IMAGE PROCESSING & ANALYSIS

IMAGE PROCESSING & ANALYSIS

Image segmentation

Basic concepts

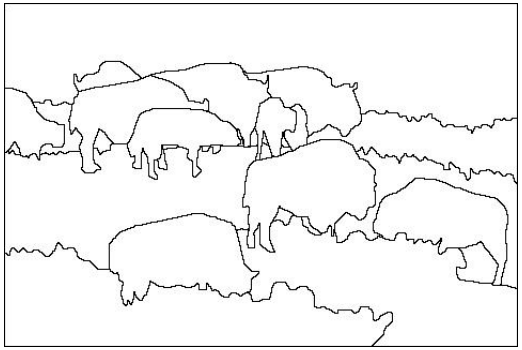
Goals and Difficulties

- The **goal of segmentation** is:
  - to find groups of pixels that "go together" (Szeliski)
  - to partition an image into a set of disjoint regions;
    the union of all the regions must correspond to the whole image.
- The result of segmentation - the segments –
  must have some meaning for the concrete application:
  - **Object Recognition**: separate the objects from the background
  - **Visual Inspection**: detect defects
  - **Robotic Vision**: locate objects / road boundaries
  - **Road Traffic Control**: detect moving objects
  - **Medical Imaging**: isolate organs;
    separate pathological/non-pathological regions.
- The results of segmentation are very important in
  determining the success or failure of image analysis
- Segmentation is **very a very difficult task**, in general.

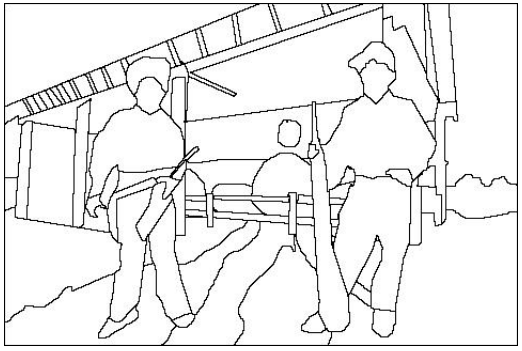
The goals of segmentation

- Separate image into coherent "objects" / "regions"
  - “Bottom-up” or “top-down” process?
  - Supervised or unsupervised?
    - use a priori knowledge or not (no training data)

image



human segmentation



Source:
S. Lazebnik

Berkeley segmentation database: http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench

The goals of segmentation

- A **region** is
  - a group of connected pixels with similar properties.
- The **goal** of segmentation is
  - to divide the image into regions,
    each of which is homogeneous in some sense,
    but the union of no two adjacent regions is
    homogeneous in the same sense.

Increasing accuracy and robustness

- Control the environment
  - e.g.: lighting
  - e.g.: in industrial applications
- Select type of sensors
  to enhance the objects of interest
  - e.g.: use infrared imaging
    for certain target recognition applications
- Introduce enough knowledge
  about the application domain
  - e.g.: size, shape or color of objects

Segmentation approaches

- **Edge-based** approaches:
  - Use the boundaries of regions to segment the image
  - Detect abrupt changes in intensity (discontinuities)
  - Gaps in edges have to be bridged
- **Region-based** approaches:
  - Use similarity and spatial proximity among pixels to find different regions

- Edges are found based on DIFFERENCES between values of adjacent pixels.
- Regions are found based on SIMILARITIES between values of adjacent pixels.

- Theoretically, both approaches should give identical results but this is not true in practice

Classifications – Unsupervised techniques (1)

Classification often not unique

- Pixel-based
  - Only based on pixel intensities
  - Only applicable in simple cases or as a preprocessing step
- Region-based
  - Allows definition of region properties
  - Takes connectivity properties into account
  - Results in connected components
- Edge-based
  - Searching for surrounding edges (often based on gradient values)
  - Gaps in edges have to be bridged
- Model-based
  - Given model is deformed to fit region
  - Explicit or implicit representation of the model
- Atlas-based
  - Pre-segmented data is mapped onto actual image by registration

Classification - next slides (2)

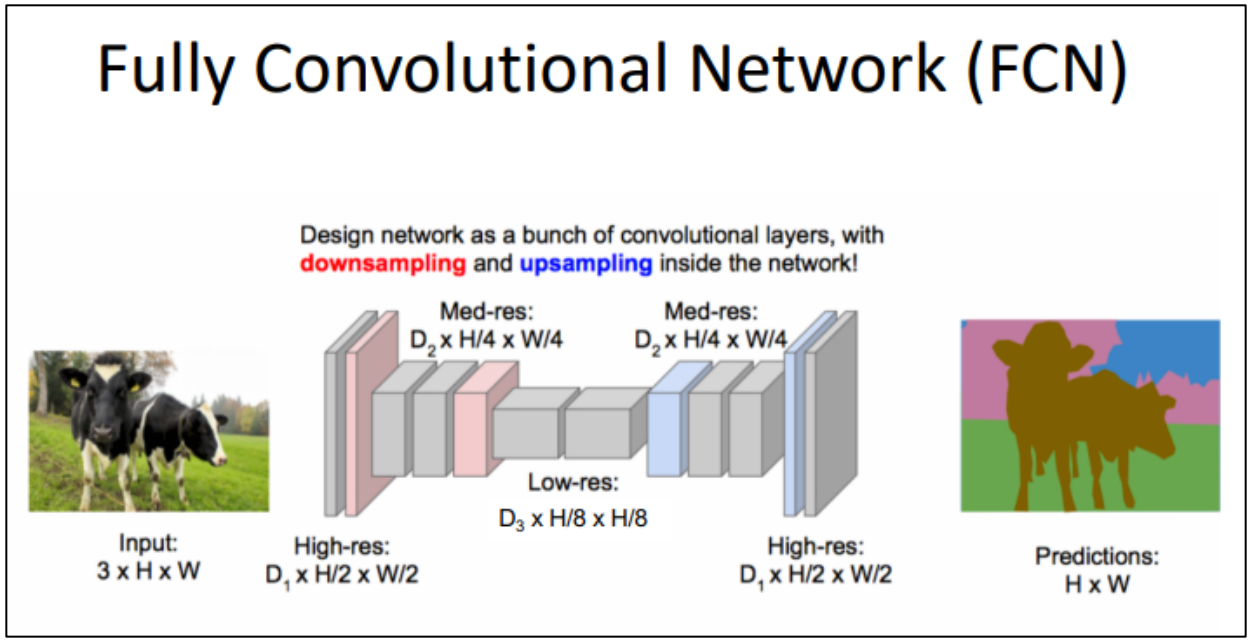
1st part of this course (next slides):

mostly

unsupervised techniques

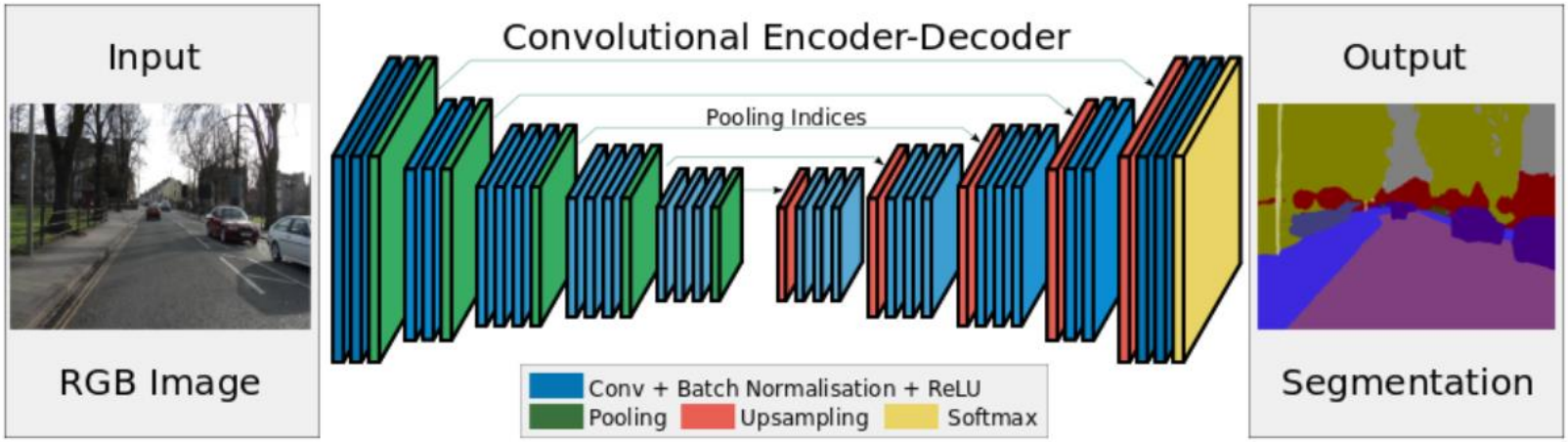
- Feature domain
  - thresholding
  - clustering
    - k-means
    - mean-shift

- Image domain
  - Boundary-based
    - edge-based
    - deformable ~~models~~ shapes
      - snakes
      - level sets
    - model-based
      - Hough transform
      - active shape models (=> training)
      - ~~atlas-based~~
  - Region-based
    - region growing
    - region splitting
    - region merging
    - split-and-merge
    - watersheds
  - Graph-based
    - livewire
    - graph cuts



To be studied in the 2nd part of the course

slide from 2nd part of the course



source: http://mi.eng.cam.ac.uk/projects/segnet/#research

Segmentation in the feature domain

- The idea is to measure several **features**, organize them into a **feature vector** and use **clustering** methods to segment the data.
- Depending on the problem, many features can be used, such as:
  - Intensity
  - Colour
  - Texture
  - Position
  - Geometrical features (size, orientation, shape …)
  - ...
- Each feature component may characterize a single pixel (as the intensity) or a region (as the texture).
- The choice of features and how they are quantified implies a **feature space** in which each token is represented by a point.
- Token similarity is thus measured by distance between points (**feature vectors**) in feature space.

IMAGE SEGMENTATION

Thresholding

Region growing/splitting

Split-and-merge

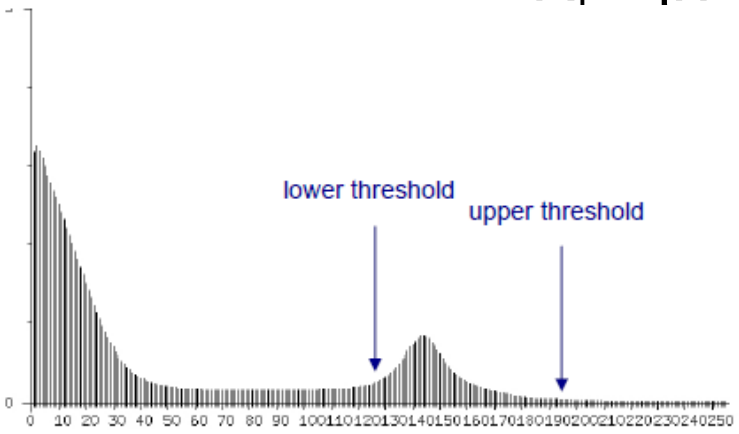
Basic segmentation algorithms

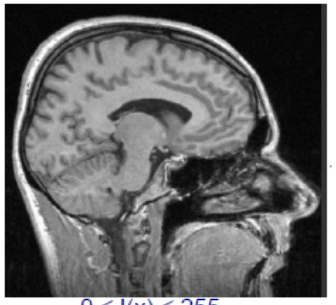
- Basic segmentation algorithms presented on the next slides:
  - Thresholding
  - Region growing
- Useful for
  - easy segmentation tasks
    - e.g.: bone in CT
  - initializing or autoseeding more sophisticated algorithms

Thresholding

- Regions with uniform intensity give rise to strong peaks in the histogram !
- Select **one or more** intensity values (**thresholds**)
- Partition the image domain according to the intensity being higher or lower than those thresholds

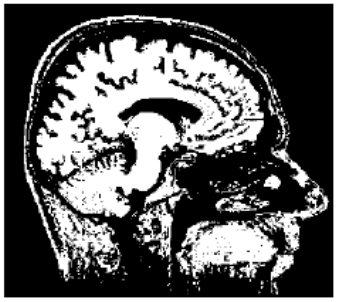
$$R_i = \{x \in D_I : t_i \leq I(x) < t_{i+1}\}$$





$0 \leq I(x) < 255$

t_i =127
t_{i+1}=195



R_i

- **Selection of thresholds**
  - manual
  - automatic
    - at local minima (requires smoothing of histogram first)
    - at intersection of fitted Gaussian functions
    - from prior knowledge
      - intensity characteristics of the objects
      - sizes of the objects
      - fractions of an image occupied by the objects
      - number of different types of objects appearing in an image
- In general, good thresholds can be selected if the histogram peaks are
  - tall,
  - narrow,
  - symmetric,
  - and separated by deep valleys.

Thresholding - automatic

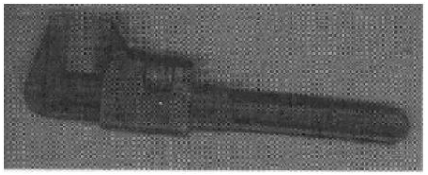
Otsu method

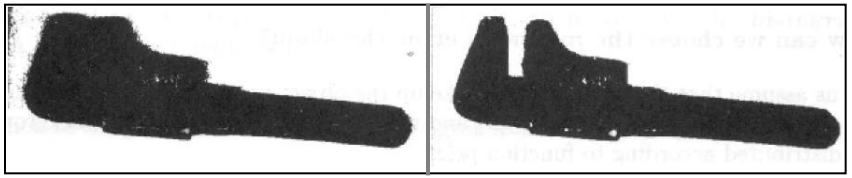
- A measure of region homogeneity is variance (i.e., regions with high homogeneity will have low variance).
- Otsu's method selects the threshold, T, such that
  - the intra-class variance is minimized and
  - the between-class variance is maximized .
- Iterative procedure
  - => calculate between-class variance for every possible T
- It does not depend on modeling the probability density functions, however, it assumes a bimodal distribution of gray-level values (i.e., if the image approximately fits this constraint, it will do a good job).

- Drawbacks:
  - Assumes that the histogram of the image is bimodal (i.e., two classes).
  - Breaks down … when the two classes are very unequal (i.e., the classes have very different sizes).

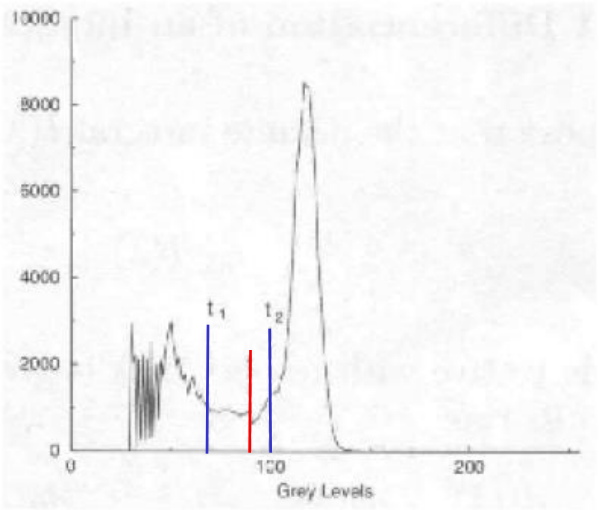
Hysteresis thresholding

- If there is no clear valley in the histogram of an image, it means that there are several background pixels that have similar gray level value to object pixels and vice-versa.
- Two thresholds, one at each side of the valley can be used in this case.
- Pixels above the high threshold are classified as **object** and below the low threshold as **background**.
- Pixels between the low and high thresholds are classified as object only if they are adjacent to other object pixels.



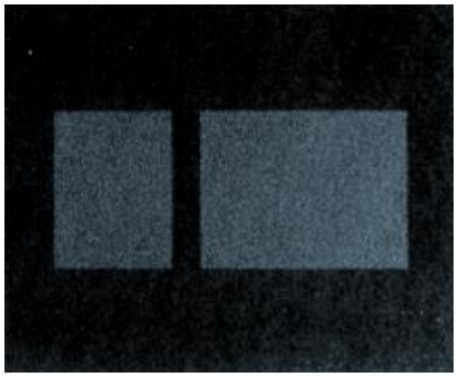


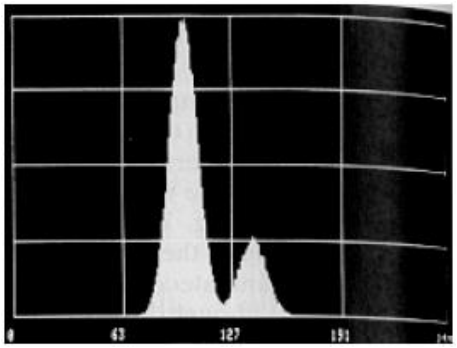
Single thresholding Hysteresis thresholding



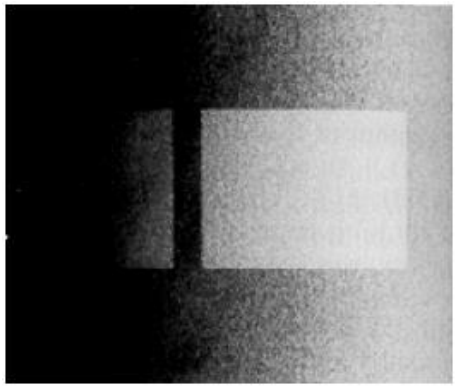
Effect of illumination on segmentation

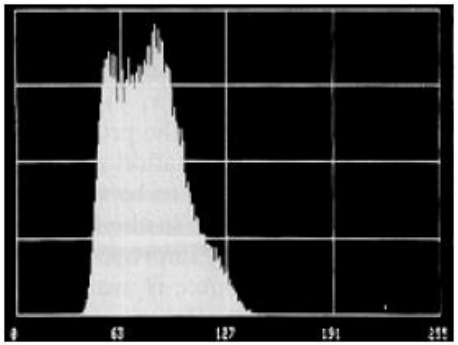
- A single threshold will not work well when we have **uneven illumination** due to shadows or due to the direction of illumination.





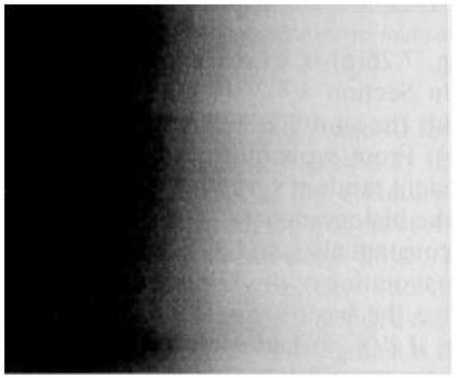
non-uniform illumination





ATTENUATING THE PROBLEM:

1) Obtain an image of just the illumination field and …



2) Make illumination uniform whenever possible

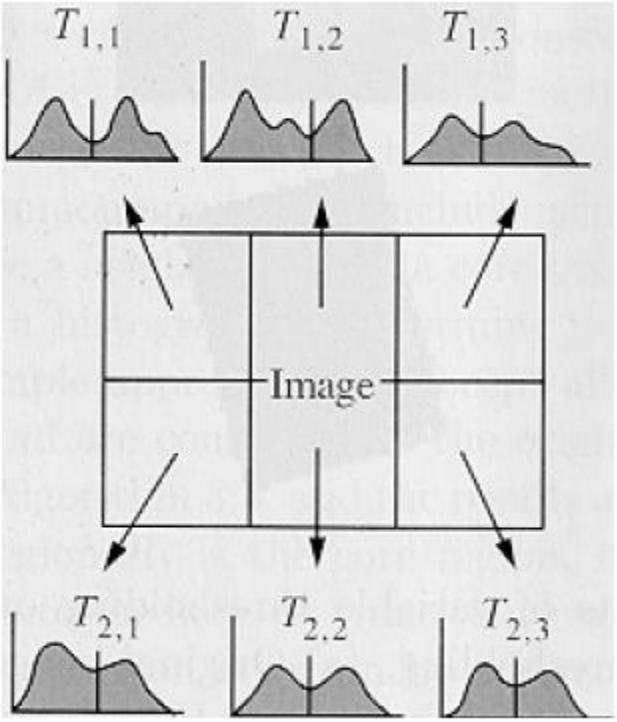
3) Local thresholding →

Effect of illumination on segmentation

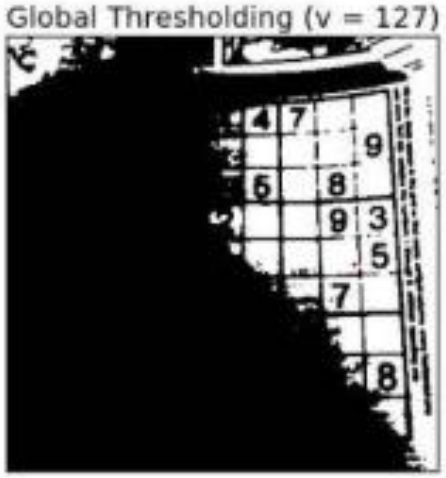
Handling nonuniform illumination: local thresholding

- The idea is to
  - partition the image into $m \times m$ subimages and
  - then choose a threshold $T_{i,j}$ for each subimage.
- This approach might lead to subimages having simpler histogram (e.g., bimodal)

- Drawback:
  - object contours not continuous in the transition between regions



OpenCV thresholding



cv2.threshold
cv2.adaptiveThreshold
etc.

Adaptive Mean Thresholding
- threshold value is
the mean of neighbourhood area

Adaptive GaussianThresholding
- threshold value is the
weighted sum of
neighbourhood values
where weights are gaussian

http://docs.opencv.org/master/d7/d4d/tutorial_py_thresholding.html#gsc.tab=0

Drawbacks of thresholding:

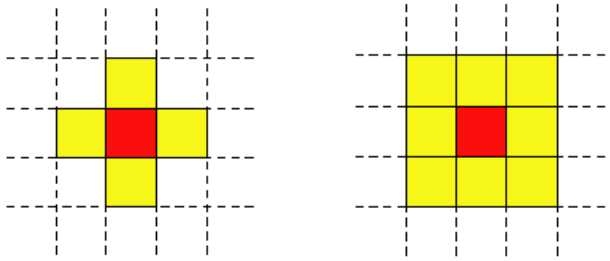
- Pixels assigned to a single class
  need not form coherent regions
  as the spatial locations of pixels are completely ignored
  - only hysteresis thresholding
    considers some form of spatial proximity.
- Threshold selection is not always straightforward.

Region growing

- Region-growing approaches exploit the important fact that pixels which are close together have similar gray values.

- General algorithm:
  - Start with a single pixel (seed) and add new pixels iteratively
  1. Choose the seed pixel
  2. Check the neighboring pixels and add them to the region if they are similar to the seed
  3. Repeat step 2 for each of the newly added pixels; stop if no more pixels can be added.

- Crucial definitions:
  - Neighborhood
  - Homogeneity (aggregation criterion)
  - Stopping criterion

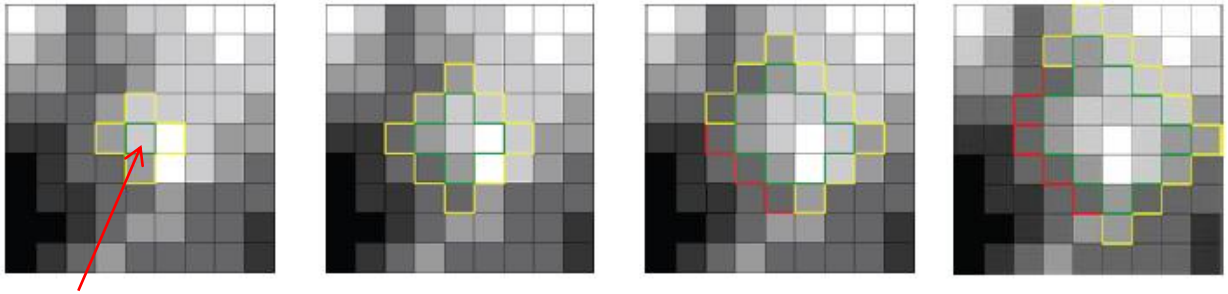


Neighborhood

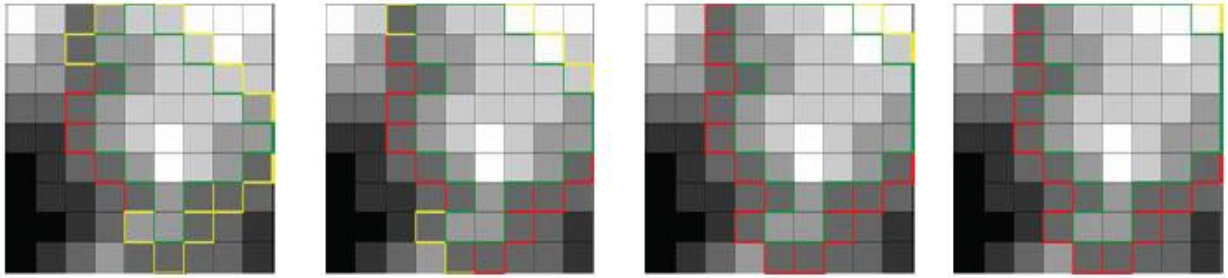
Region growing

- How to choose the seed(s) in practice ?
  - It depends on the nature of the problem.
    - e.g.: if targets need to be detected using infrared images, choose the brightest pixel(s).
  - Without a-priori knowledge, compute the histogram and choose the gray-level values corresponding to the strongest peaks
- How to choose the similarity criteria (predicates)?
  - The homogeneity predicate can be based on any characteristic of the regions in the image such as
    - average intensity
    - variance
    - color
    - texture
    - shape
    - ...

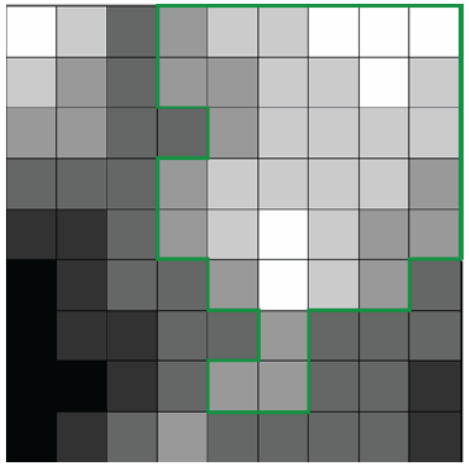
Region growing



seed point



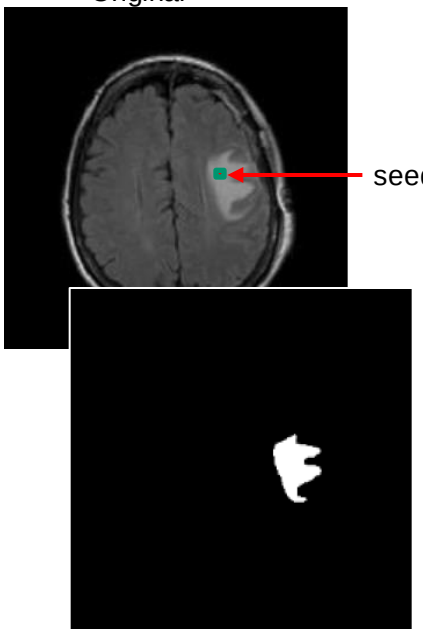
Similarity criterion: gray value difference to seed point <= 1, neighborhood = 4



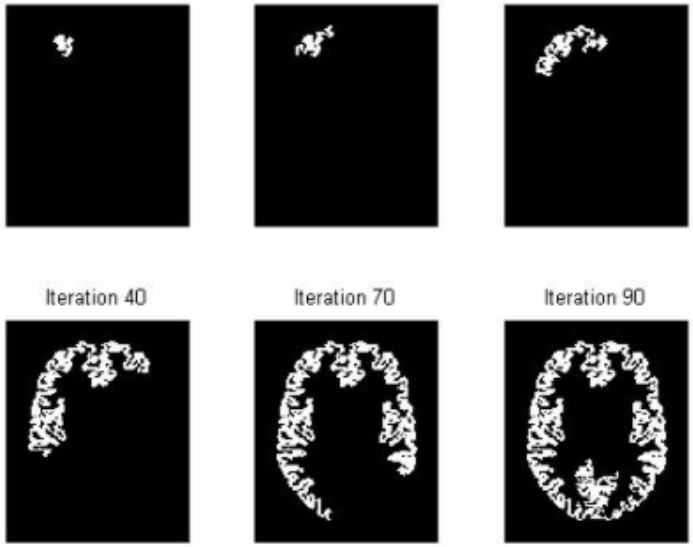
Segmentation result

Region growing

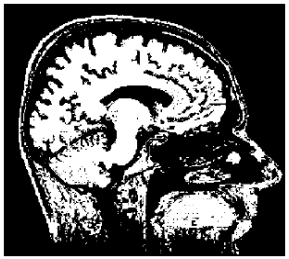
Original



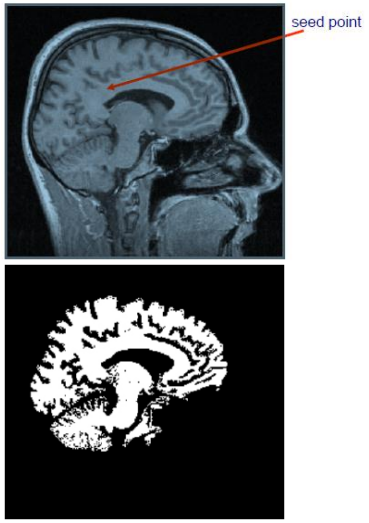
Region growing



Region growing

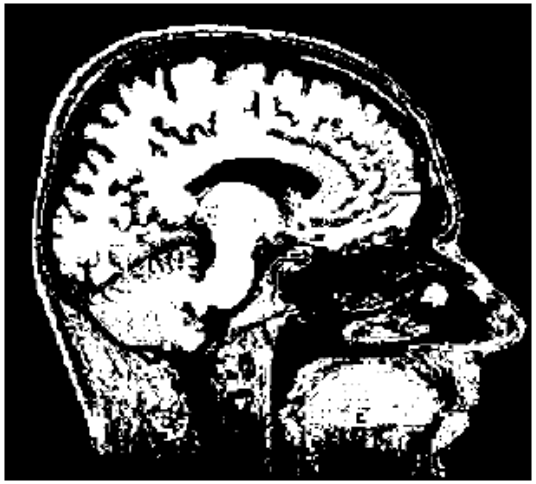


Thresholding

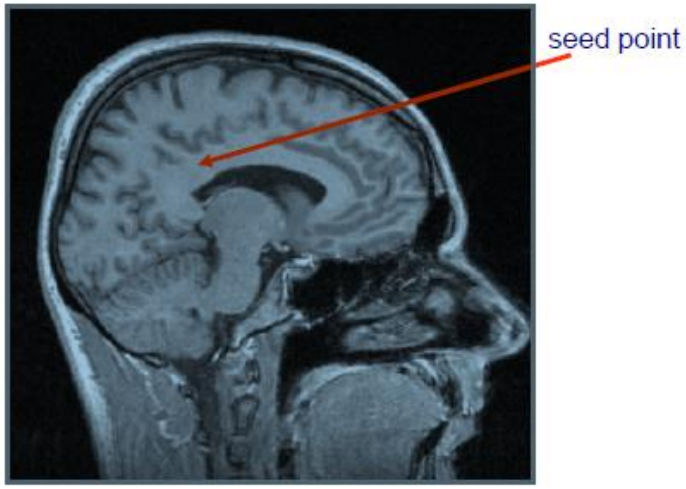


Region growing;
gray value difference = 40

Region growing vs. Thresholding



Thresholding



Region growing
Gray value difference = 40

Region Merging / Splitting

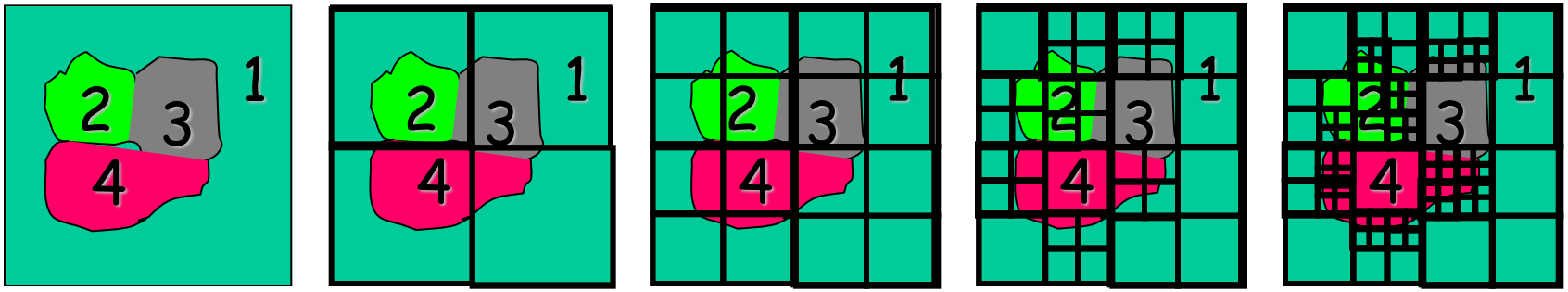
- **Region Merging:**
  - Image is separated into very small regions (like 2x2 squares, or even pixels)
  - Neighboring regions are merged
    if the merged region remains homogeneous
    or
    if the regions are similar
  - What does "similar" mean?
    - "similar" average values : $|\mu_i - \mu_j| < T_\mu$
    - "small" spread of gray values: $|I_{max} - I_{min}| < T_I$
    - surface fitting
      - model regions as polynomial surfaces
      - compute the fitting error
      - merge regions if the fitting error is low enough
    - other ...?

Region Merging / Splitting

- **Region Splitting:**
  - Whole image is one region
  - Homogeneity is computed for each region
  - Regions are successively split, until homogeneity is fulfilled for every region
  - Problem: over-segmentation
- Deciding when to split is fairly straight-forward
  - A property is not “constant”
  - A predicate is not TRUE
- Where to split is the difficulty
  - Some approaches:
    - Look for strong edges to create boundaries.
    - Divide it into equal parts along image dimensions.

Split-and-Merge

Split …



…and merge

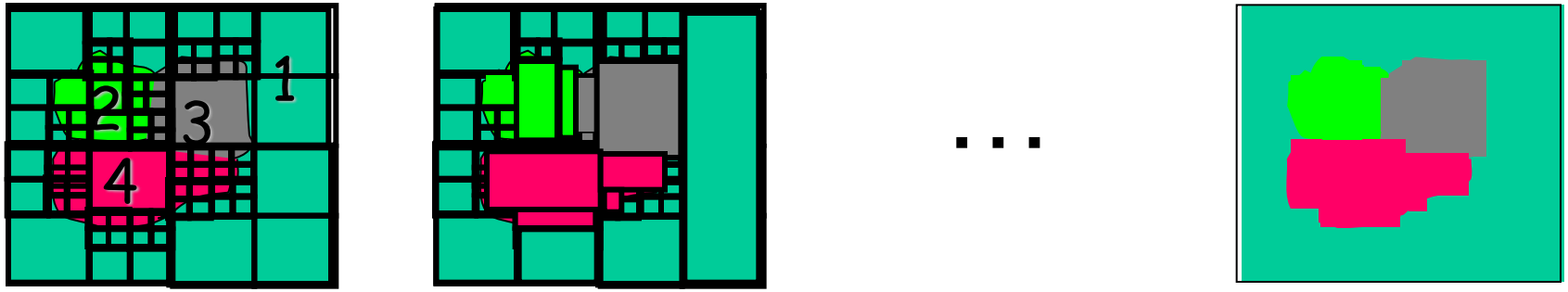
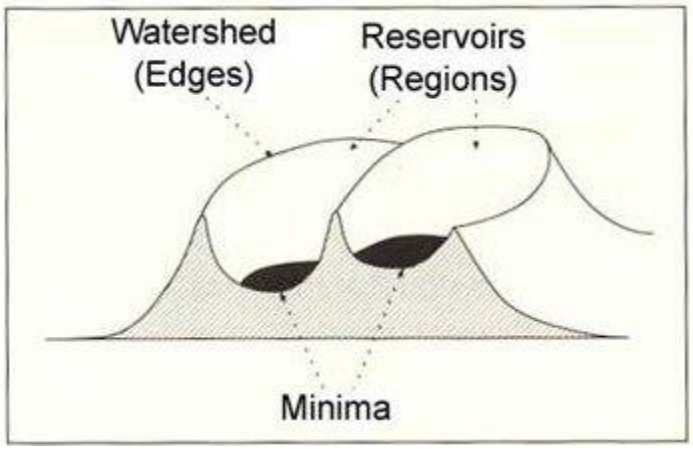


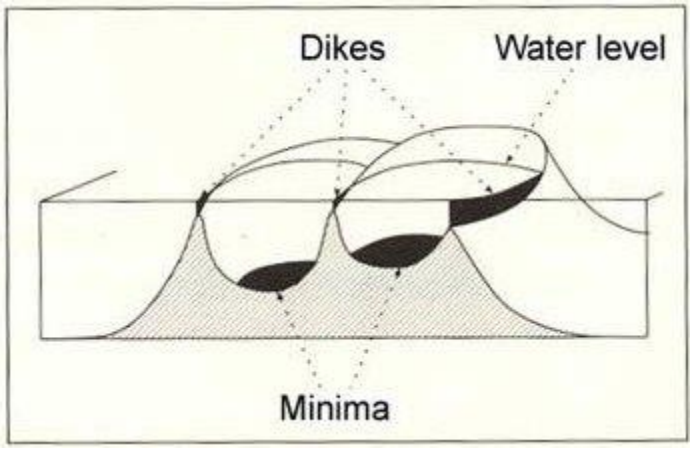
IMAGE SEGMENTATION

Watersheds

Watersheds: idea

- Interpretation of image as height map
- Height map consists of basins and ridges
- Idea: dropping water onto gray value height map
- Flooding the “landscape” would fuse regions which can be prevented by building dikes
- Result: one region for every local minimum
- Often applied to gradient magnitude images

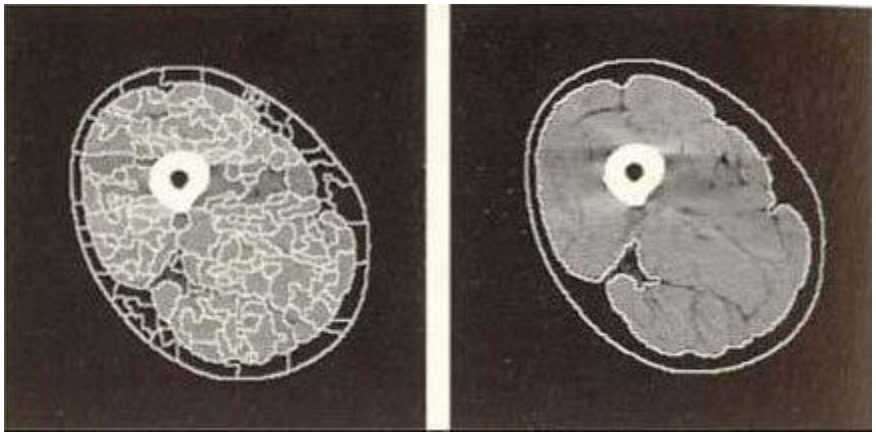




http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_watershed/py_watershed.html
http://cmm.ensmp.fr/~beucher/wtshed.html **STOP – visit these pages**

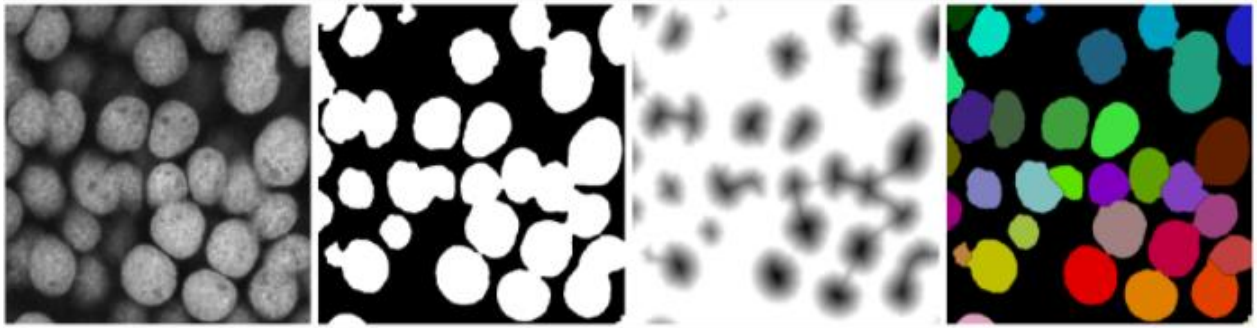
Watersheds: properties

- Boundaries are always closed around regions
- One region for every local minimum
- Raw watershed segmentation may produce a severely oversegmented image with hundreds or thousands of catchment basins.
  - Pre-processing: smoothing to prevent over-segmentation
  - Post-processing: region merging; edge information; ...
- Other possibilities to prevent over segmentation:
  - Minimum basin depth
  - Minimum basin size



Watersheds

- A classic way of separating touching objects in binary images makes use of the distance transform and the watershed method.
- The idea is to create a border as far as possible from the center of the overlapping objects. This strategy works very well on rounded objects and it is called **Distance Transform Watershed**.
- It consists on
  - calculating the distance transform of the binary image, inverting it (so the darkest parts of the image are the centers of the objects)
  - and then applying watershed on it using the original image as mask (see figure below).

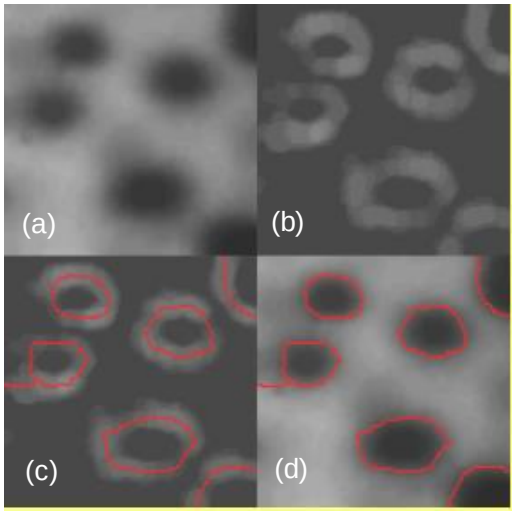


a) b) c) d)

a) Image of touching cell nuclei from a confocal laser scanning microscope
b) binary mask calculated after filtering and thresholding input image;
c) inverse of the distance transform applied to the binary mask (Chamfer distance);
d) resulting labeled image after applying watershed to the inverse distance image using the binary mask.

https://imagej.net/Distance_Transform_Watershed

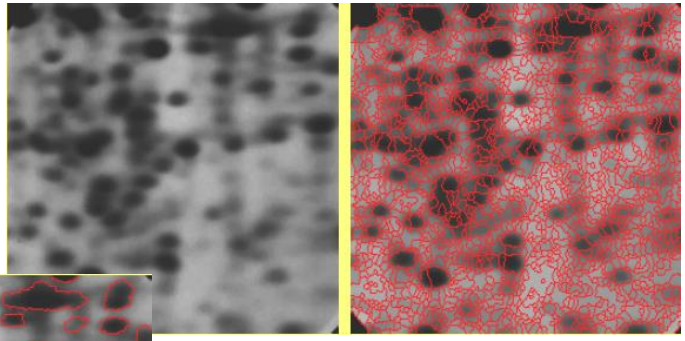
Watersheds

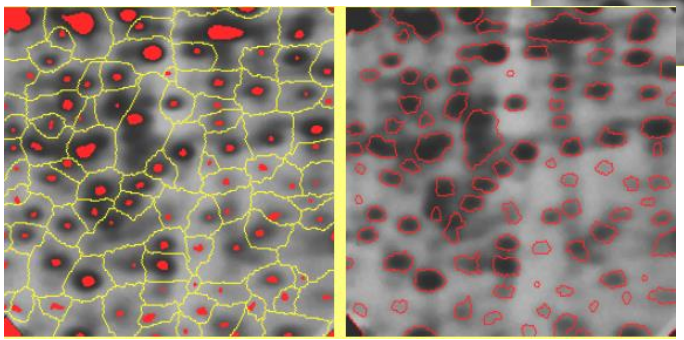


(a) - Original image.
(b) - Gradient image.
(c) - Watershed of the gradient image.
(d) - Final contours (over original image).

Electrophoresis gel image

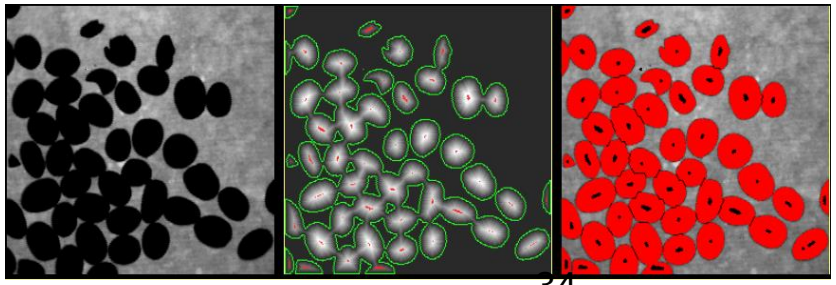
Over-segmentation due to noise or local irregularities in the gradient image





Marker-controlled watershed consists in flooding the topographic surface from a previously defined set of markers, preventing over-segmentation

Coffee beans separation
In this case, the criterion used is not the contrast (which is irrelevant) but the distance function of the initial image



https://www.cmm.mines-paristech.fr/~beucher/wtshed.html

- **void watershed(InputArray image, InputOutputArray markers)**
- Performs a marker-based image segmentation using the watershed algrorithm.
- Parameters:
- image – Input 8-bit 3-channel image.
- **markers** – Input/output 32-bit single-channel image (map) of markers. It should have the same size as image.
- The function implements one of the variants of watershed, non-parametric marker-based segmentation algorithm, described in [Meyer92].
- Before passing the image to the function, you have to roughly outline the desired regions in the image **markers** with positive (>0) indices. So, every region is represented as one or more connected components with the pixel values 1, 2, 3, and so on.
- Such markers can be retrieved from a binary mask using **findContours**() and **drawContours**() *(see the watershed.cpp demo)*.
- The markers are “seeds” of the future image regions. All the other pixels in markers, whose relation to the outlined regions is not known and should be defined by the algorithm, should be set to 0’s.
- In the function output, each pixel in markers is set to a value of the “seed” components or to -1 at boundaries between the regions.

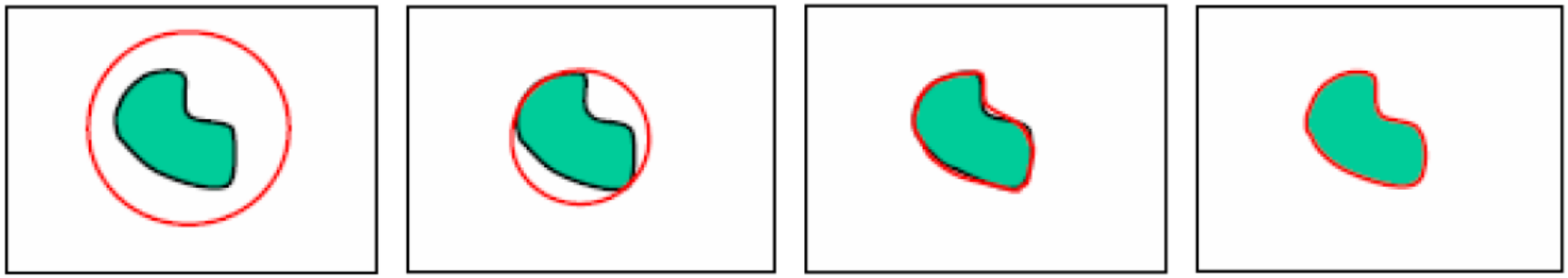
Active contours

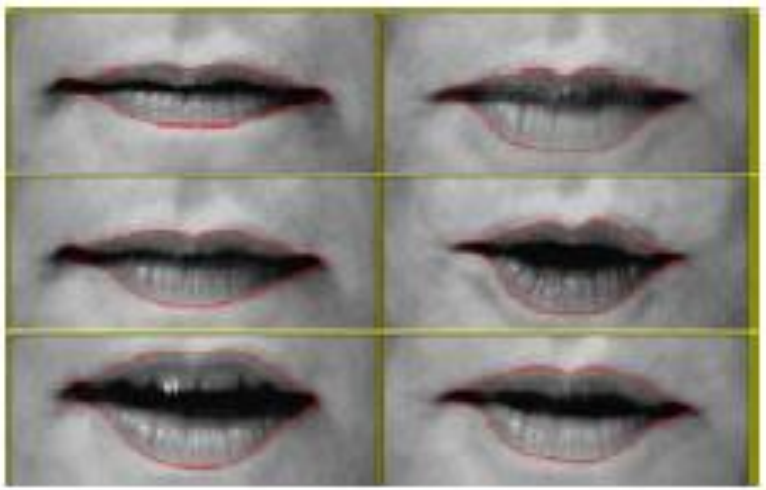
Active contours:

- Snakes
  - An energy-minimizing, two-dimensional spline curve that evolves (moves) towards image features such as strong edges - *(Kass, Witkin, and Terzopoulos, 1988)*
- Level sets
  - The central idea is to represent the evolving contour using a signed function whose zero corresponds to the actual contour. i.e., region boundaries are implicitly represented as the zero level set of a level set function
  - Overcomes some of the limitations of snakes.
- Active Shape Models
  - A *point distribution model* that describes the average contour of a class of objects and the allowable displacement of the points the contour.
  - The *point distribution model* is aligned with image object by searching for image edge points along the normal to the contour.
- Intelligent scissors
  - As the user draws a rough outline, the system computes and draws a better curve that clings to high-contrast edges, in real-time – *(Mortensen and Barrett,1995)*

Snakes

- Segmentation using a deformable model which represents the interface between background and object
- The snake is a contour represented parametrically as $c(s) = (x(s), y(s))$ where x(s) and y(s) are the coordinates along the contour and $s \in [0,1]$
- Snake minimizes energy composed of *(see Trucco book)*
  - internal energy:
    - only depends on snake itself, not on image intensities
    - usually curvature-dependant leading to some regularization
  - external energy:
    - image-dependant
    - usually gradient-dependant, attracting snake to edges
  - Possibly some "balloon force" expanding or shrinking the model
- Stationary solution for the minimization problem gives desired segmentation
- User-interaction:
  - drawing initial boundary curve
- Comments:
  - This approach is simple and has low computational requirements.
  - It does not guarantee convergence to the global minimum of the energy functional.
  - Works very well as far as the initial snake is not too far from the desired solution.





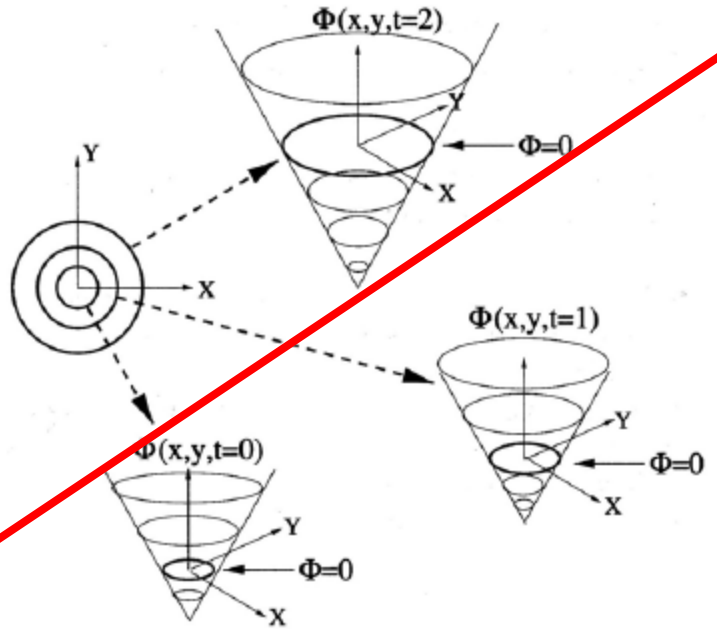
Source: Szeliski book

Lip tracking

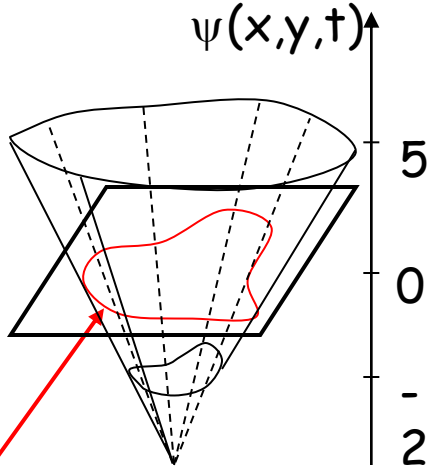
see also https://scikit-image.org/docs/stable/auto_examples/edges/plot_active_contours.html

Level sets

- Level sets
  - Drawbacks of snakes:
    - fixed topology
      (number of connected components and possible holes)
    - self-intersection possible
  - Solution:
    - Implicit representation of interfaces or boundaries (curves in 2D, surfaces in 3D)



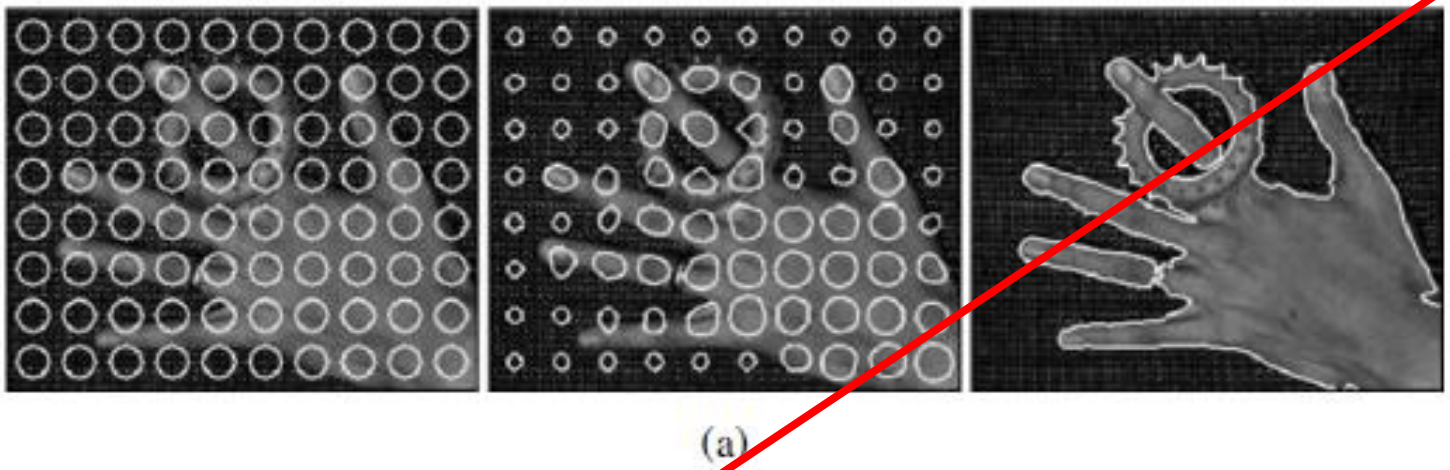
7	6	5	4	4	4	3	2	1	1	1	2	3	4	5
6	5	4	3	3	3	2	1	0	0	0	1	2	3	4
5	4	3	2	2	2	1	0	-1	-1	-1	0	1	2	3
4	3	2	1	1	1	0	-1	-2	-2	-2	-1	0	1	2
3	2	1	0	0	0	-1	-2	-3	-3	-2	-1	0	1	2
2	1	0	-1	-1	-1	-2	-3	-3	-2	-1	0	1	2	3
2	1	0	-1	-2	-2	-3	-3	-2	-1	0	1	2	3	4
2	1	0	-1	-2	-2	-2	-2	-1	0	1	2	3	4	5
3	2	1	0	-1	-1	-1	-1	-1	0	1	2	3	4	5
4	3	2	1	0	0	0	0	-1	-1	0	1	2	3	4
5	4	3	2	1	1	1	1	0	0	1	2	3	4	5
6	5	4	3	2	2	2	2	1	1	2	3	4	5	6



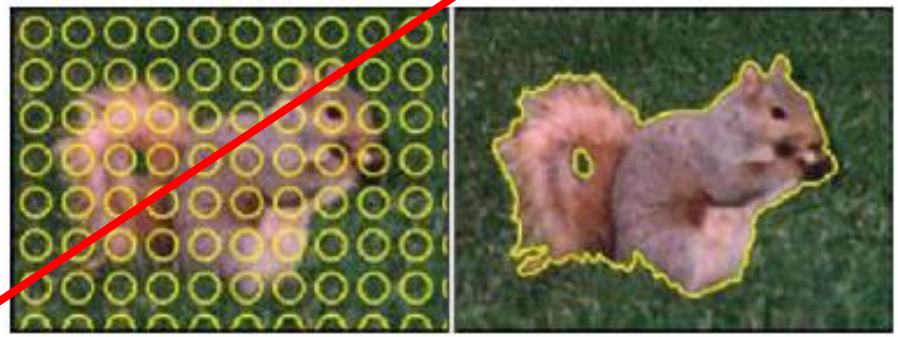
ψ(x,y,t)

γ(t)

Level sets



(a)



(b)

Source: Szeliski book

Level set segmentation (Cremers, Rousson, and Deriche 2007)
(a) grayscale and (b) color image segmentation.
The initial circles evolve towards an accurate segmentation of foreground and background, adapting their topology as they evolve.

Active Shape Models (ASM's)

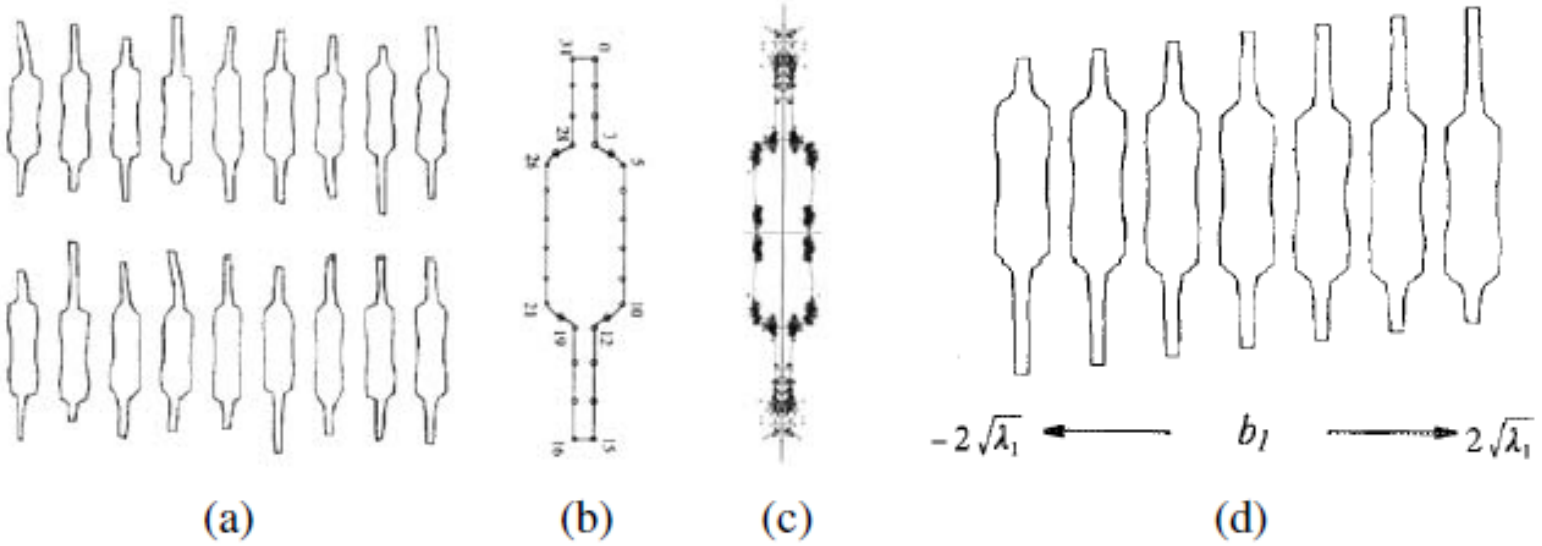
- **Active shape models** (ASM's)
  - are statistical models of the shape of objects which iteratively deform to fit to an example of the object in a new image.
  - The shapes are constrained by the PDM (point distribution model) Statistical Shape Model to vary only in ways seen in a training set of labelled examples.
  - The shape of an object is represented by a set of points (controlled by the shape model).
  - The ASM algorithm aims to match the model to a new image.
    It works by alternating the following steps:
    - Look in the image around each point for a better position for that point
    - Update the model parameters to best match to these new found positions
  - To locate a better position for each point one can:
    - look for strong edges, or
    - a match to a statistical model of what is expected at the point.

- Application examples:
  - analyse images of faces,
  - medical image segmentation (in 2D and 3D; ex: lung segmentation).

- It is also known as a **"Smart Snakes" method**,
  since it is analog to an active contour model which would respect explicit shape constraints
- It is closely related to the **Active Appearance Model**.

- *T.F. Cootes and C.J. Taylor and D.H. Cooper and J. Graham (1995).*
  "Active shape models – their training and application".
  Computer Vision and Image Understanding (61): 38–59.

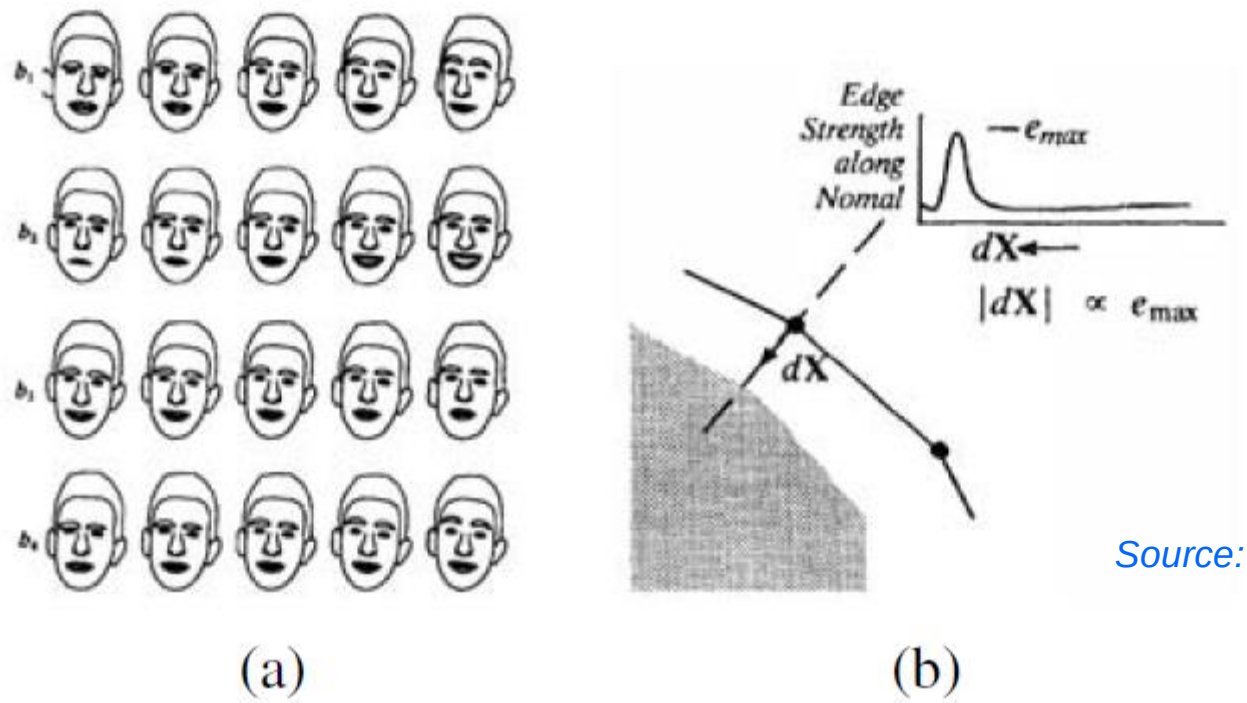
Active Shape Models (ASM's)



Point distribution model for a set of resistors (Cootes, Cooper, Taylor et al.1995)
(a) set of input resistor shapes;
(b) assignment of **control points** to the boundary;
(c) distribution (scatter plot) of point locations;
(d) first (largest) mode of variation in the ensemble shapes: λ_1.

Source: Szeliski book

Active Shape Models (ASM's)

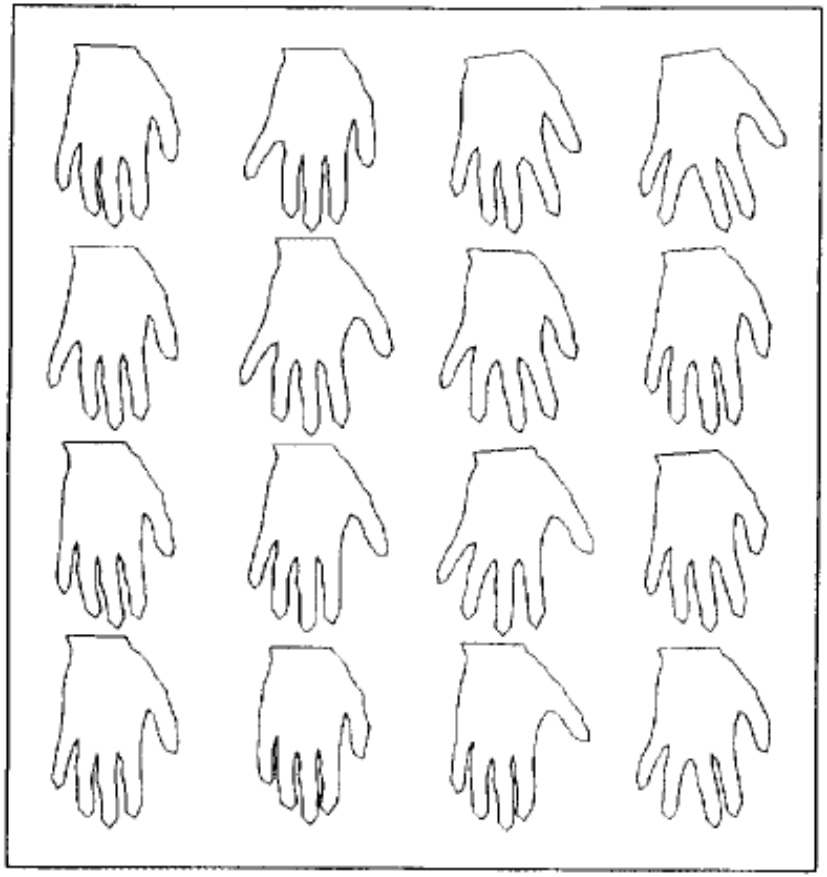


Source: Szeliski book

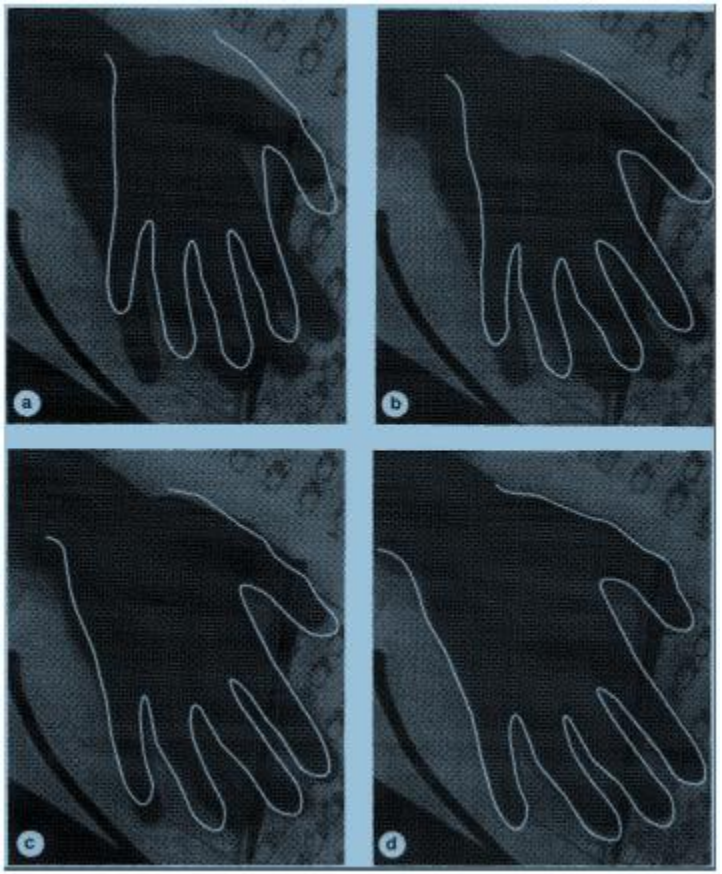
Active Shape Model - ASM (Cootes, Taylor, Lanitis et al. 1993):
(a) the effect of **varying the first four shape parameters** of a set of faces
(b) searching for the strongest gradient along the normal to each control point

UoMASM - A C++ Active Shape Model Library from the University of Manchester
https://sourceforge.net/projects/uomasm/

Active Shape Models (ASM's)



Training set of hand shapes each defined by 72 points

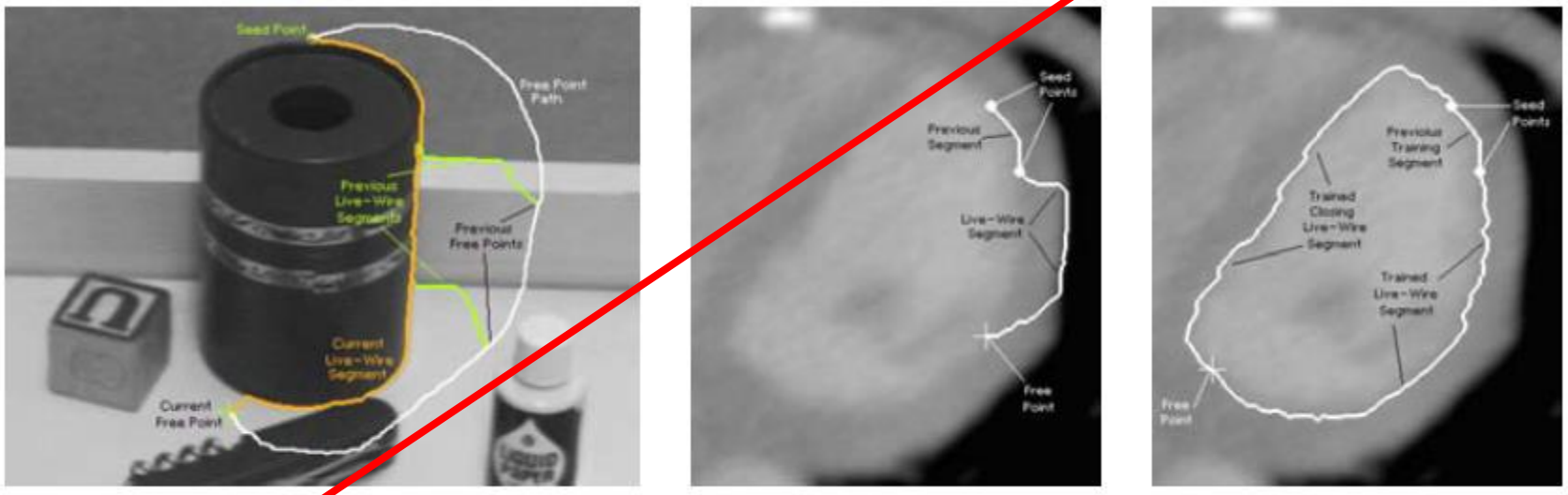


Hand with hand model superimposed; initial position and location after 100, 200 and 350 iterations

Source:
T.F. Cootes and C.J. Taylor and D.H. Cooper and J. Graham (1995)."Active shape models – their training and application". Computer Vision and Image Understanding (61): 38–59

Intelligent scissors / Live-wire

- Approach answers a basic question
  - Q: how to find a path **from** seed **to** mouse
    that follows object boundary as closely as possible ?
  - A: Instead of connecting user-selected anchor points with straight lines,
    a minimum cost path is chosen.
    Low costs can be found along pixels that exhibit strong edge features.



(a) (b) (c)

(a) as the mouse traces the white path, the scissors follow the orange path along the object boundary (the green curves show intermediate positions) (Mortensen and Barrett,1995)
(b) regular scissors can sometimes jump to a stronger (incorrect) boundary
(c) after training to the previous segment, similar edge profiles are preferred (Mortensen and Barrett,1998)

IMAGE SEGMENTATION

Clustering

Clustering

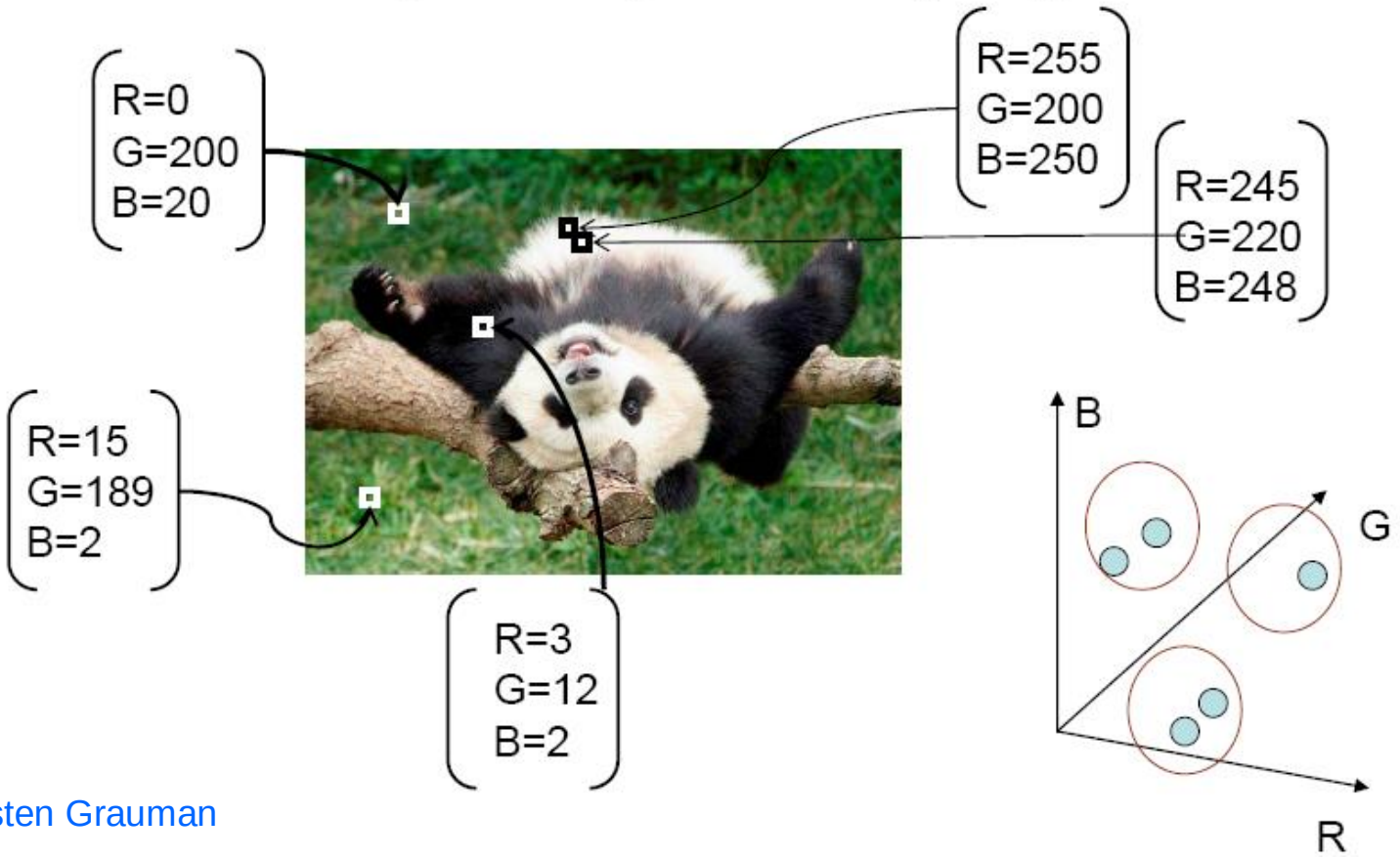
- Clustering:
  - classification methodology for grouping the **feature space** into a set of meaningful groups or classes.
    - *unsupervised classification*
      - *no "training data"*
      - *model inference and application both rely on the "test data" exclusively*
    - *supervised classification:*
      - *use "training data" to infer model (the classes)*
      - *apply model to "test data"*

Find groups inherent to data (clustering)

Find a “classifier” for known classes
→ *2nd part of the course*

- Two main approaches:
  - **Divisive clustering** (or clustering by splitting)
    - the entire data set is regarded as cluster and the clusters are recursively split to yield a good clustering
  - **Agglomerative clustering** (or clustering by merging)
    - each data item is regarded as a cluster and clusters are recursively merged until a good set of clusters is obtained.

- Cluster similar pixels (features) together



Slide credit: Kristen Grauman

K-means clustering

K-means algorithm

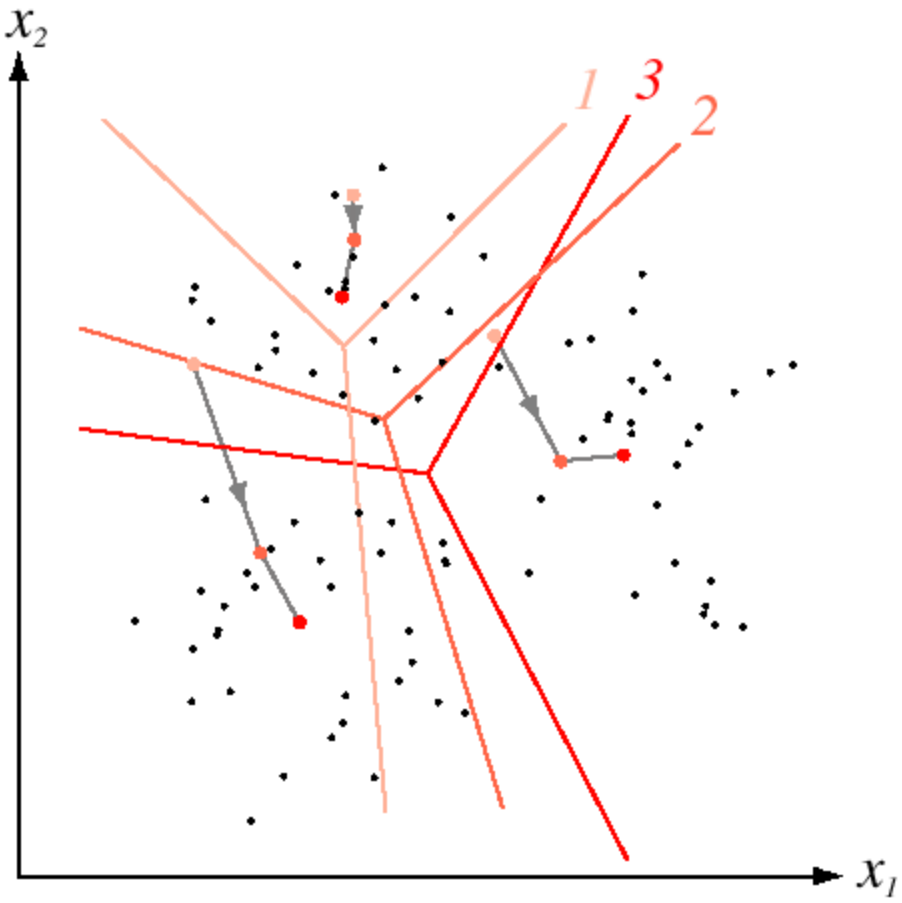
- Initialization: given
  - **N** points in **feature space**
  - **K** categories.
- Pick **K** points randomly
  - these are initial cluster centers (means): $\mu_1, \ldots, \mu_K$.
- Repeat :
  - assign each of the N points, x_j, to clusters by nearest μ_i
  - recompute mean μ_i of each cluster from its member points

  Until no mean has changed.

How to find K ?
- Use prior knowledge about image.
- Use different K's and test for goodness of clusters.
- Analyze image histograms.

K-means clustering

Example: 3-means clustering

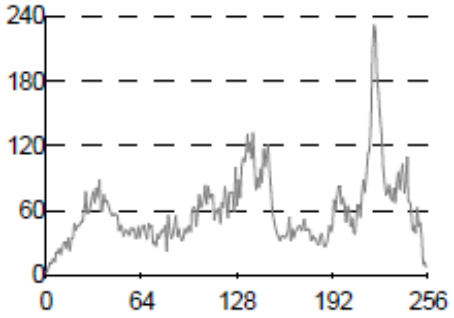


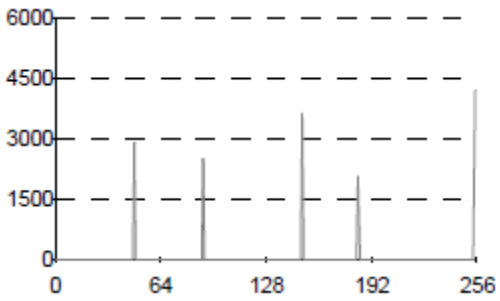
K-means clustering

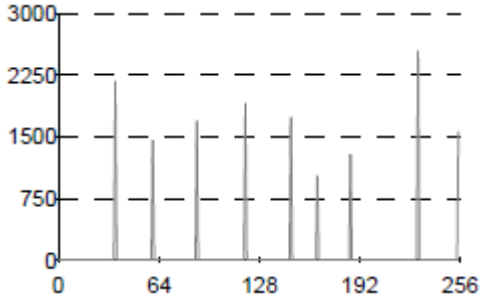
original image

K=5 clusters

K=9 clusters







NOTE: clusters don't have to be spatially coherent …

K-means clustering

- K-means clustering based on intensity or color
  - Clusters don't have to be spatially coherent

Image

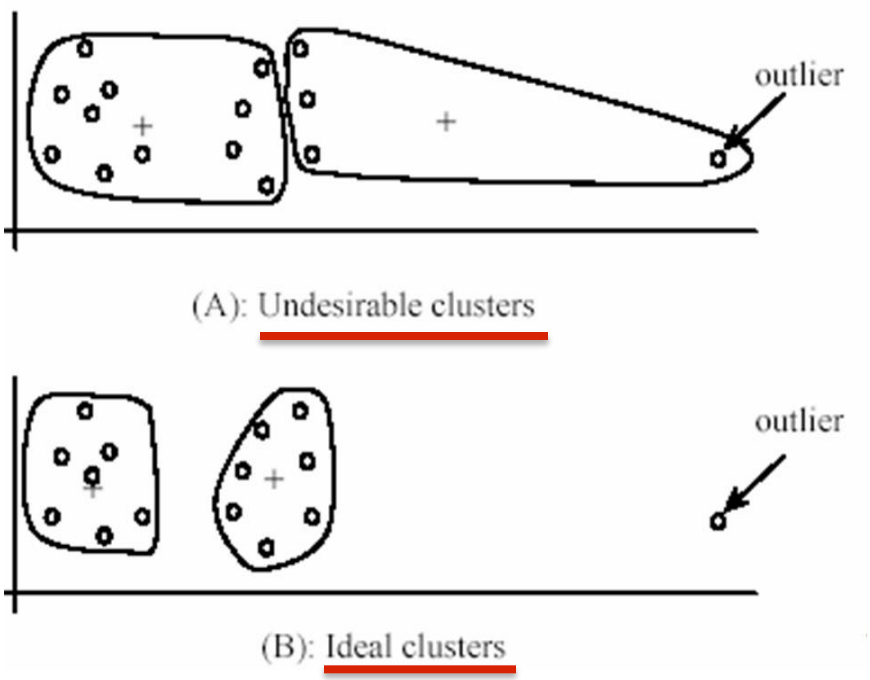
Intensity-based clusters

Color-based clusters

- **Clustering based on (color_components & position-(X,Y)** values enforces **more spatial coherence**

K-means clustering

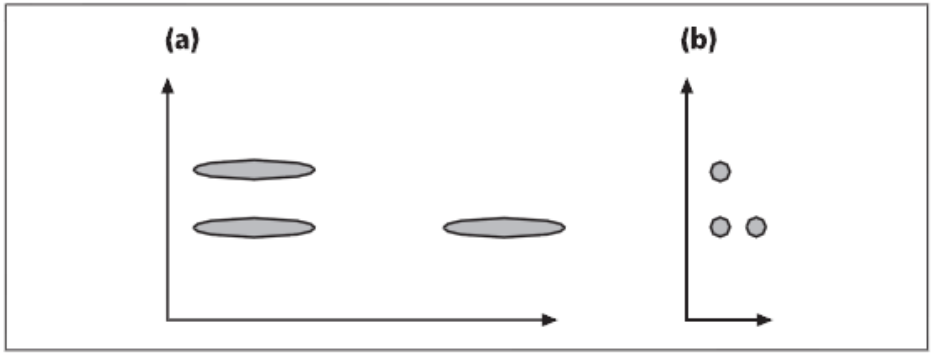
- Pros
  - Very simple method
  - Converges to a local minimum of the error function
- Cons
  - Need to pick K
  - Sensitive to initialization
  - Sensitive to outliers
  - Only finds "spherical" clusters



(Mahalanobis distance)

Mahalanobis distance:

- A distance measure that takes into account the variance of the data.
- In particular,
  distance from the mean along a direction where there is little variance has a large weight and
  distance from the mean along a direction where there is a large variance has little weight.
- If the covariance is the identity matrix (identical variance),
  then this measure is identical to the Euclidean distance measure.



The Mahalanobis computation allows us to reinterpret the data's covariance as a "stretch" of the space:

(a) the vertical distance between raw data sets is less than the horizontal distance;

(b) after the space is normalized for variance,
the horizontal distance between data sets is reduced

Superpixel algorithms

- Superpixel algorithms
  - group pixels into perceptually meaningful atomic regions, which can be used to replace the rigid structure of the pixel grid;
  - they capture image redundancy, provide a convenient primitive from which to compute image features, and can reduce the complexity of subsequent image processing tasks
  - In SLIC a single parameter specifies the number of superpixels (tricky to choose …).

SLIC - Simple Linear Iterative Clustering
adapts k-means clustering
to generate superpixels (*)

Radhakrishna Achanta, et al.,
EPFL Technical Report no. 149300,
June 2010.

Images segmented using SLIC into superpixels of size 64, 256, and 1024 pixels (approxim.)

(*) – there are other superpixel algorithms, alternative to SLIC (ex: SLICO, a zero parameter version)
OPENCV → http://docs.opencv.org/trunk/df/d6c/group__ximgproc__superpixel.html

IMAGE SEGMENTATION

Mean-shift clustering

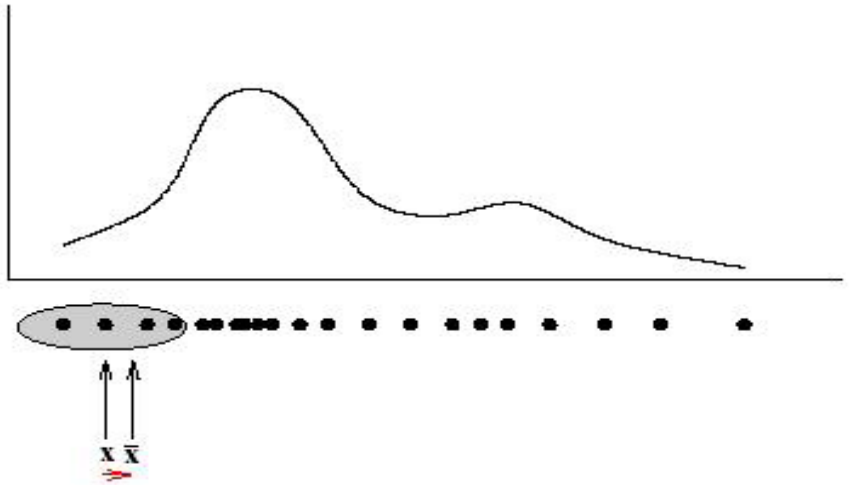
Mean-shift algorithm

- The mean shift algorithm seeks the “mode” or point of highest density of a data distribution
  - using a non-parametric technique
    - arbitrary distributions
    - without assuming forms of the underlying densities
  - based on Robust Statistics
    - "robust" is used in its formal statistical sense: that is, mean-shift ignores outliers in the data
      - this means that it ignores data points that are far away from the peaks in the data
    - it does so by processing only those points that are within a local window of the data and then moving the window
- The key to mean shift is a technique for efficiently finding peaks in a high-dimensional data distribution without ever computing the complete function explicitly

Mean-shift algorithm

- What is mean-shift ?
  - Given any density function,
    the mean of a set of samples taken about **x**
    will be biased towards a local mode (i.e. local maximum)
  - The mean-shift vector $\mathbf{\bar{x}-x}$
    points in the direction of the gradient

- Algorithm mean-shift to find histogram peak:
  1. Choose a window size
  2. Choose the initial location
     of the search window
  3. Compute the mean location
     in the search window
  4. Center the window
     at the location computed in 3
  5. Repeat steps 3 and 4
     until convergence.

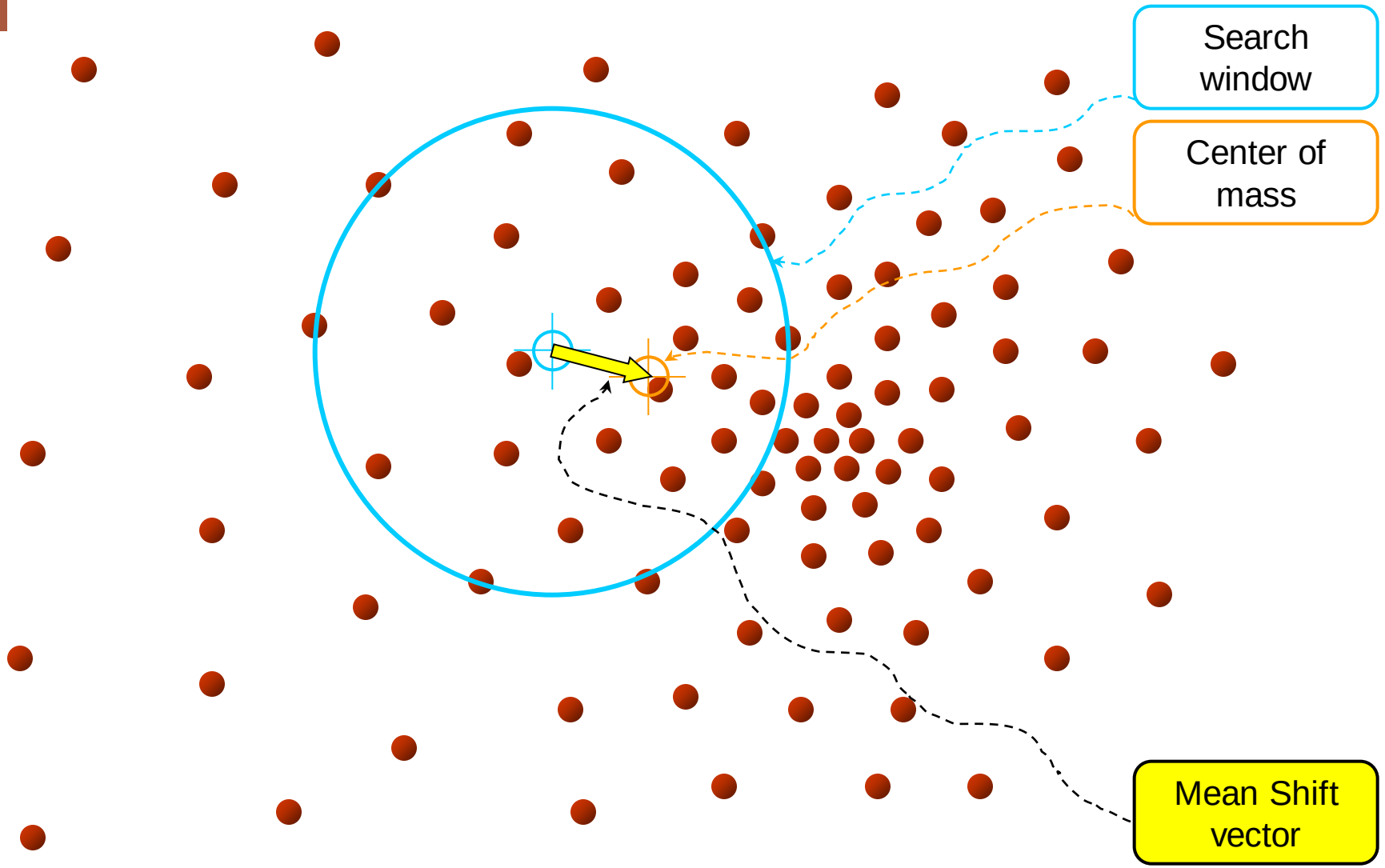


Mean-shift algorithm

1. Choose a search window
    1. its initial location
    2. its type
        - uniform, polynomial, exponential, or Gaussian
    3. its shape
        - symmetric or skewed, possibly rotated, rounded or rectangular
    4. its size
        - extent at which it rolls off or is cut off
2. Compute the window's (possibly weighted) center of mass
3. Center the window at the center of mass
4. Return to step 2 until the window stops moving

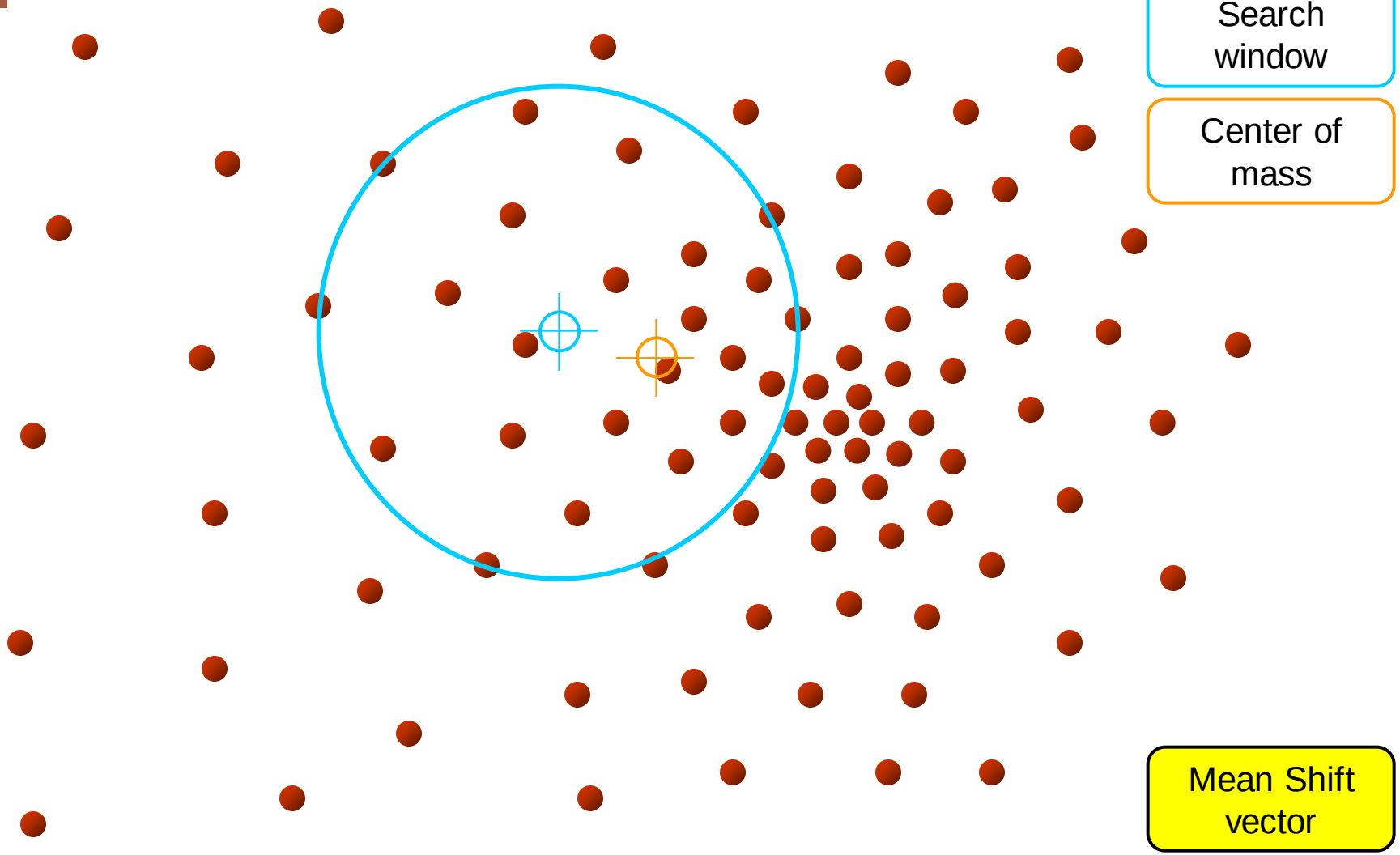
source: Bradski book

Mean-shift algorithm

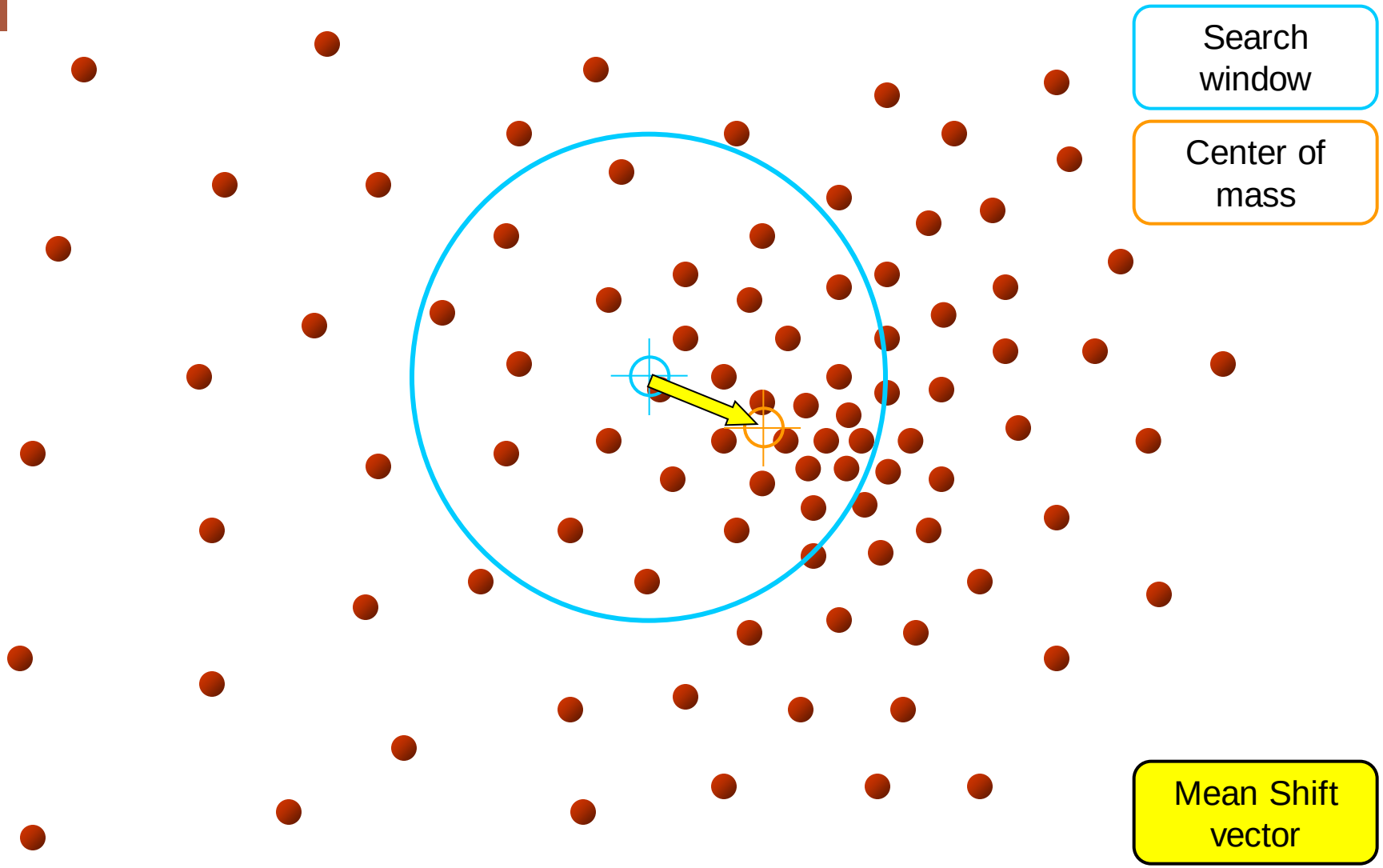


Source: Y. Ukrainitz & B. Sarel

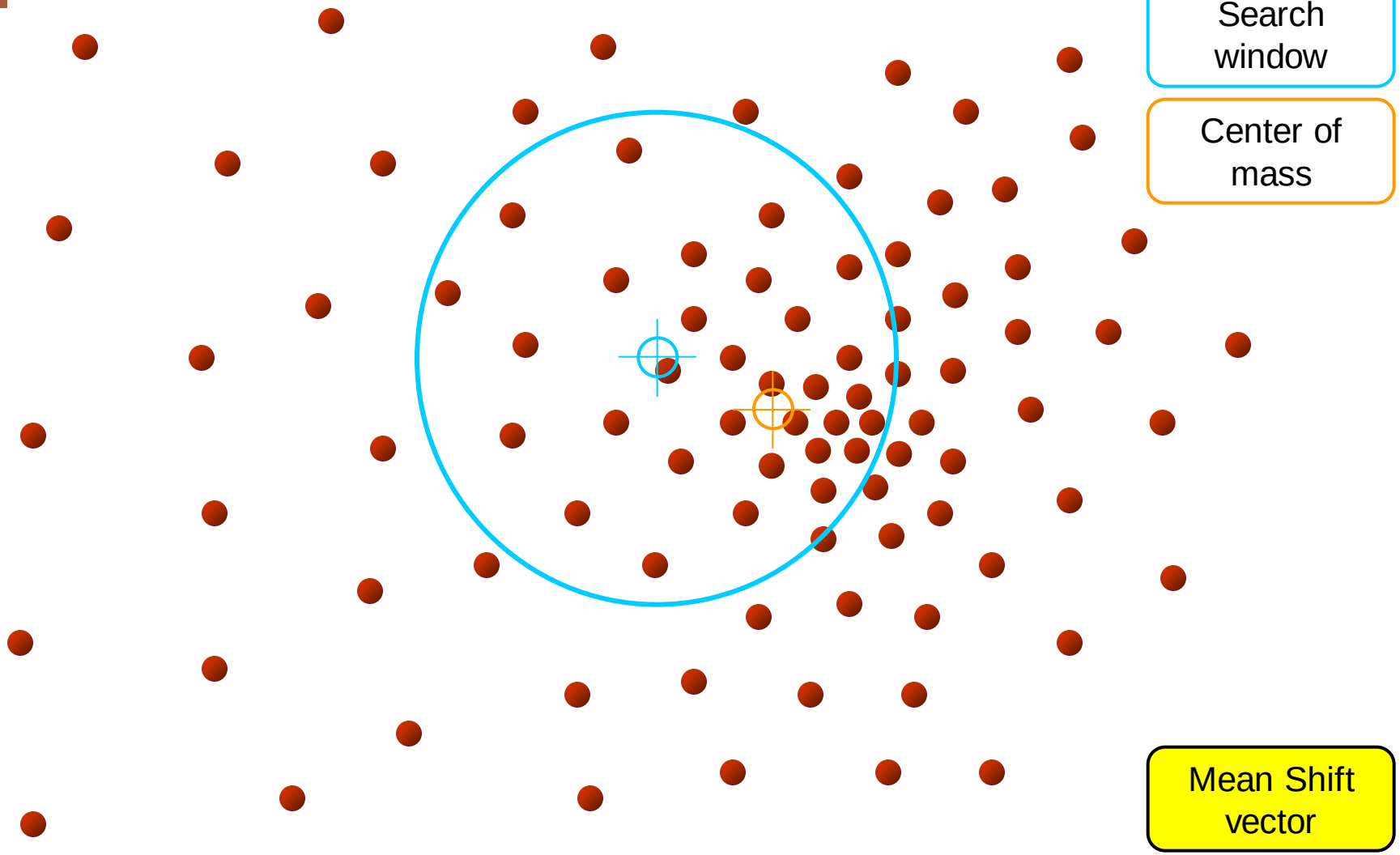
Mean-shift algorithm



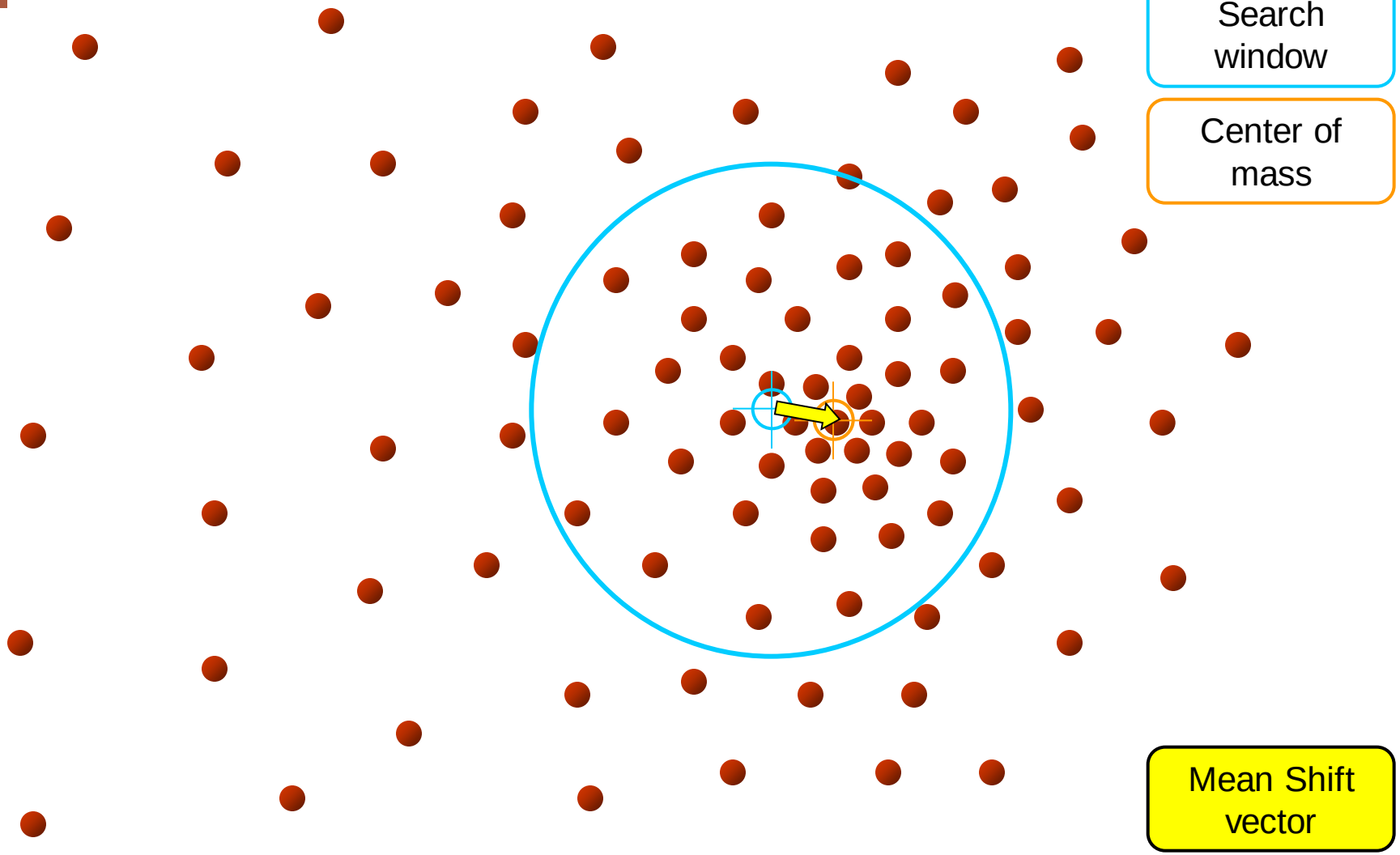
Mean-shift algorithm



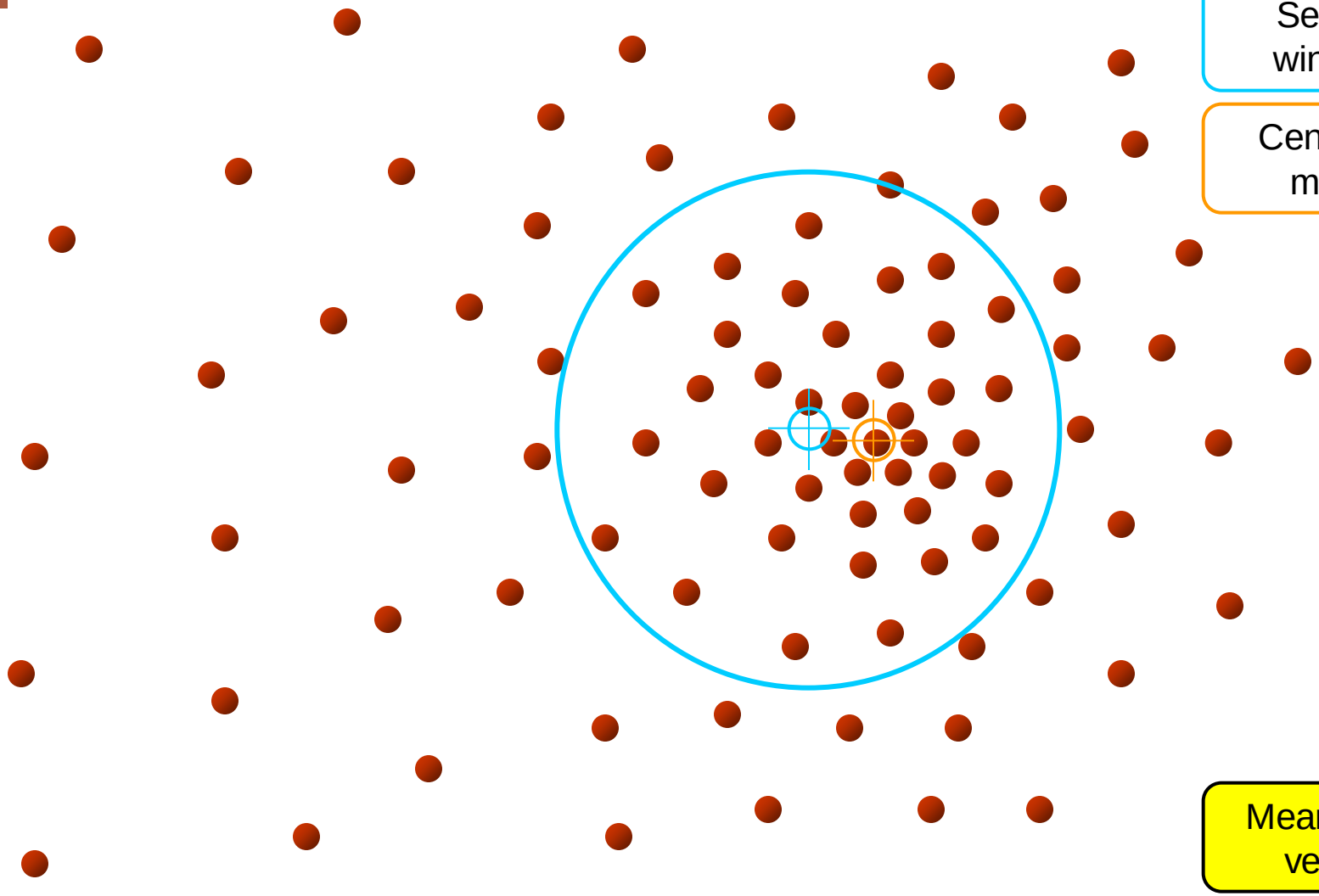
Mean-shift algorithm



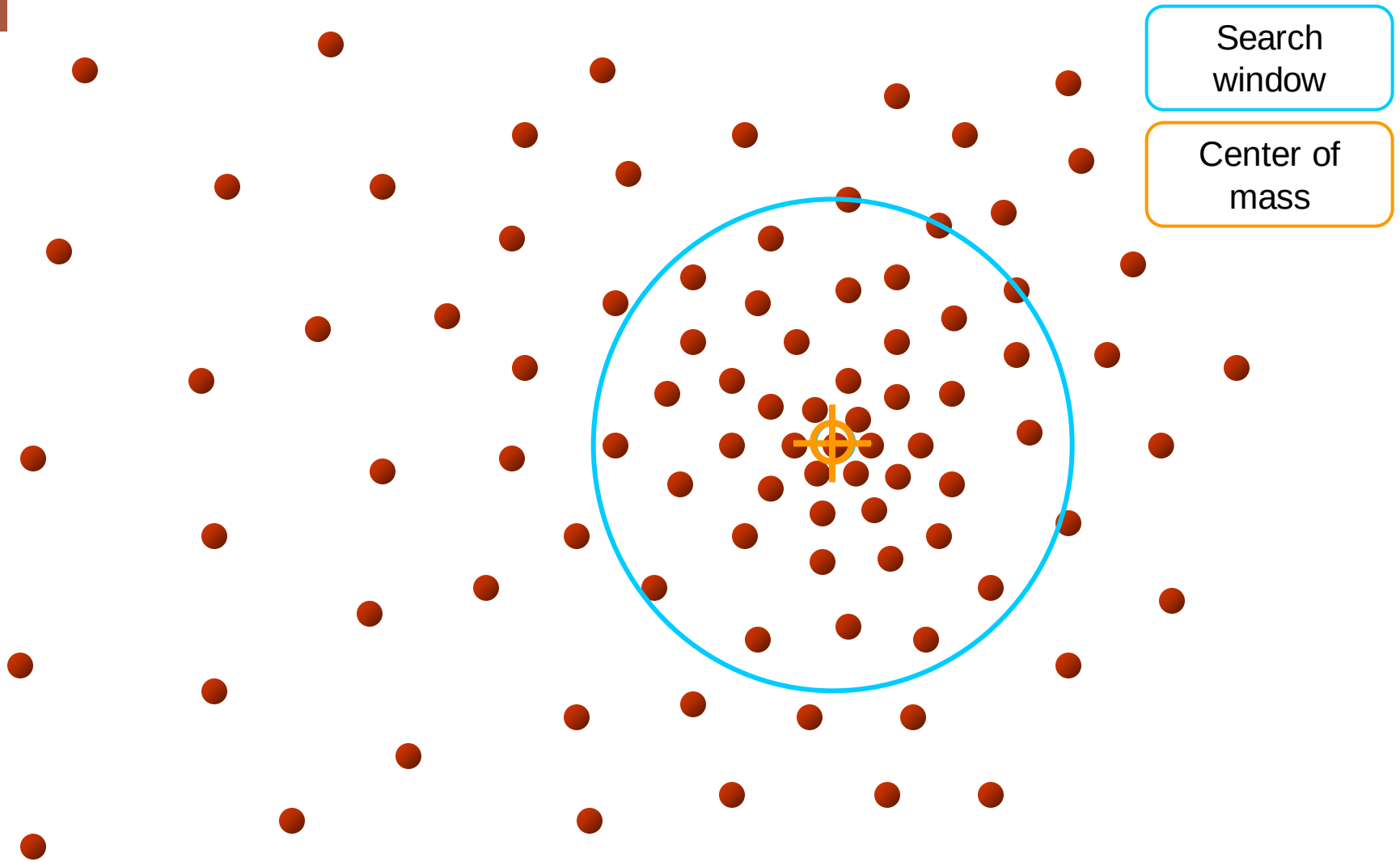
Mean-shift algorithm



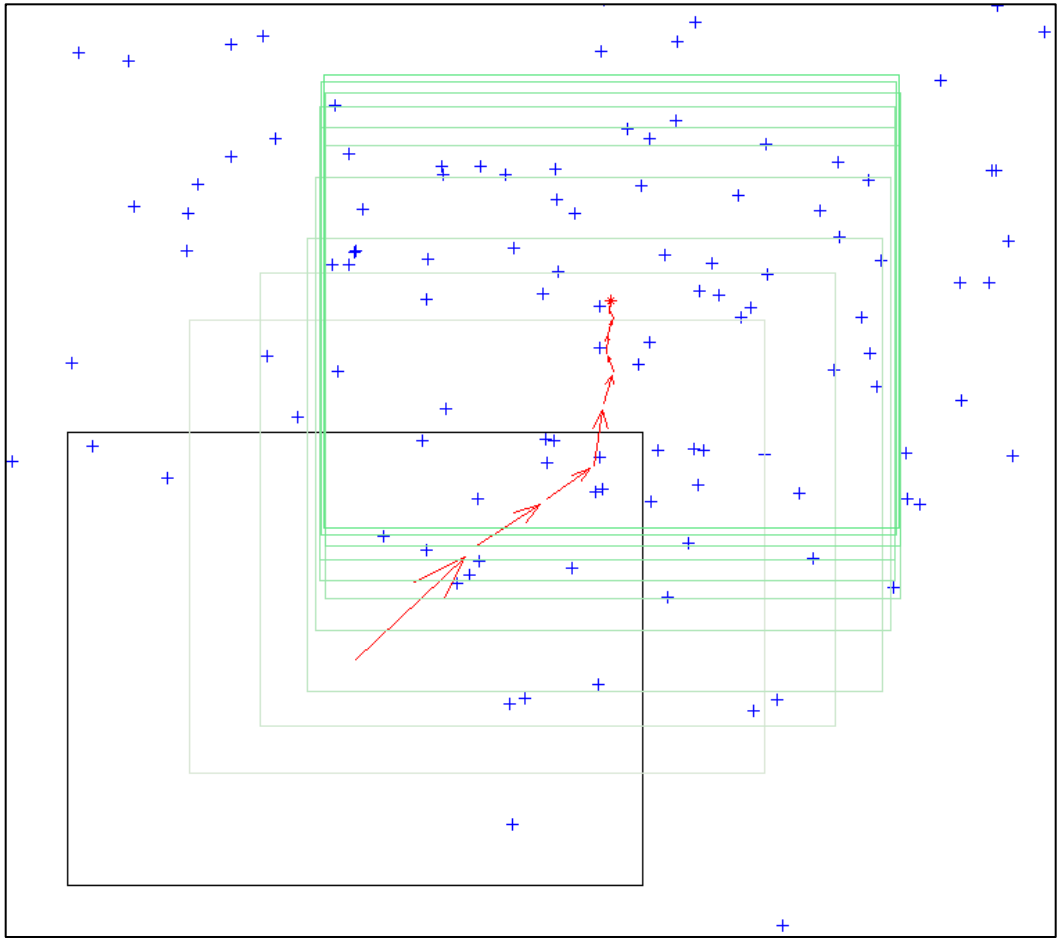
Mean-shift algorithm



Mean-shift algorithm



Mean-shift algorithm

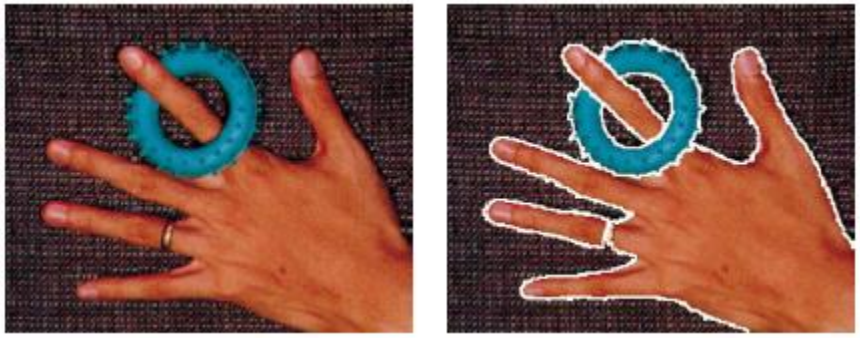


Mean-shift algorithm in action: an initial window is placed over a 2D array of data points and is successively recentered over the mode (or local peak) of its data distribution until convergence

Meanshift applications

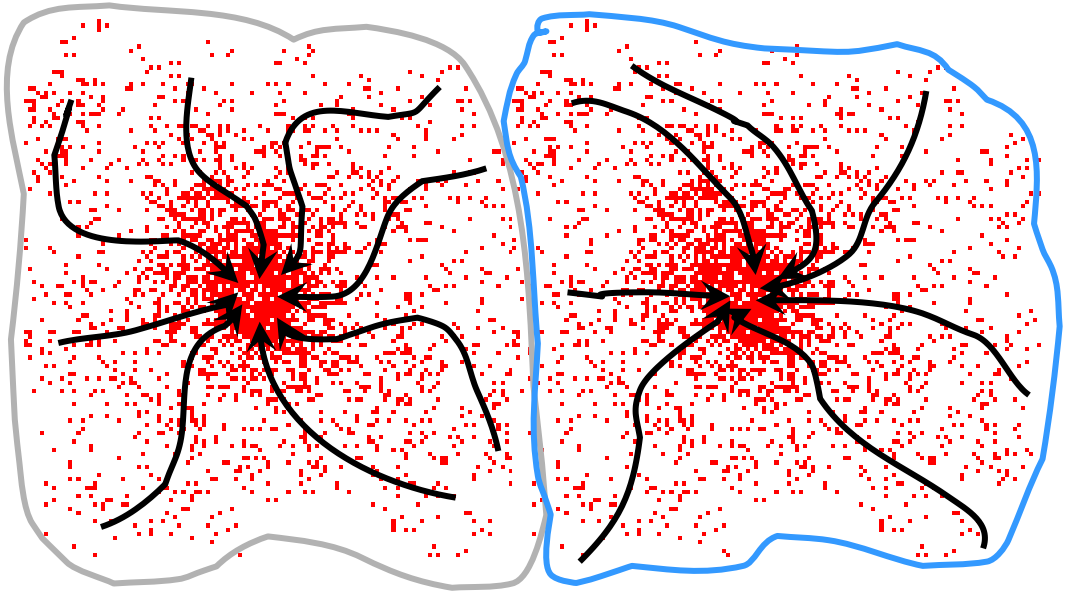
- Meanshift can be used for
  - Image smoothing
  - Image segmentation
  - Object tracking

- **Mean-shift for IMAGE SEGMENTATION**
  - Convert the image into tokens (via color, gradients, texture measures, etc).
  - Choose initial search window locations uniformly in the data.
    - *you could place a window at each data point ... (heavy!)*
  - Compute the mean shift window location for each initial position.
  - Merge windows that end up on the same “peak” or mode.
  - The data these merged windows traversed are clustered together.

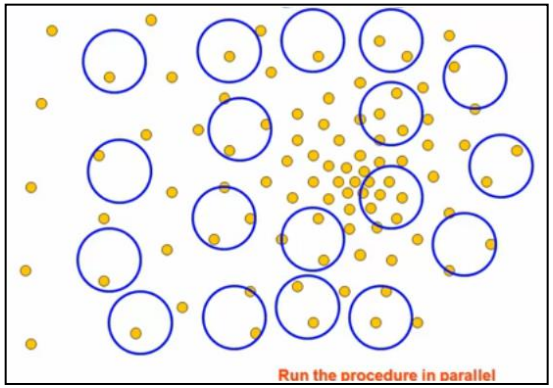


Mean-shift color image segmentation, Comaniciu and Meer 2002

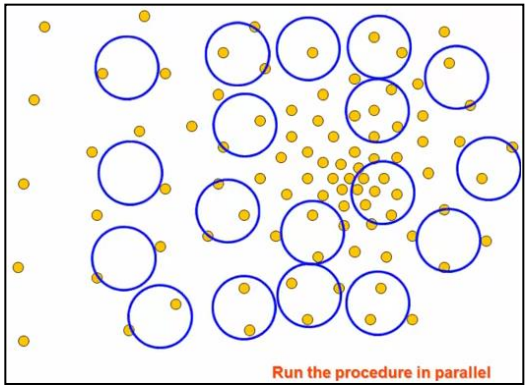
- Cluster:
  - all data points in the attraction basin of a mode
- Attraction basin:
  - the region for which all trajectories lead to the same mode



- Window initialization and peak finding

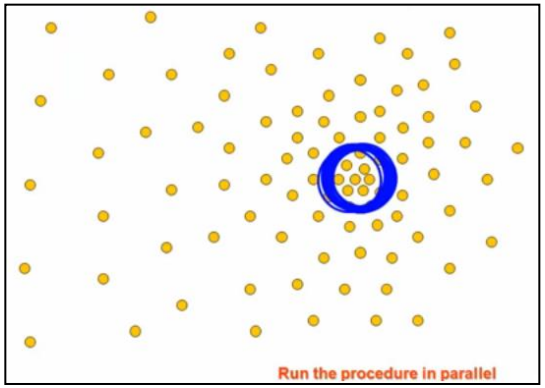


Initialization
(*could place a window at each data point ... (heavy!)*

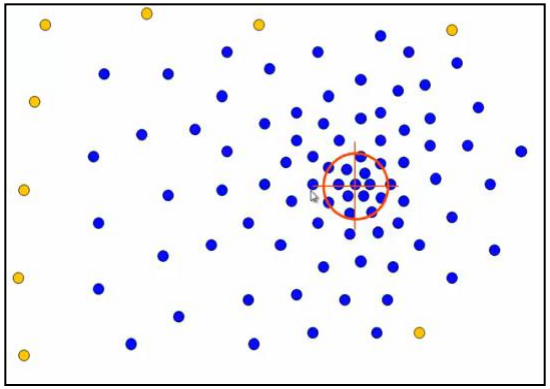


Movement

...

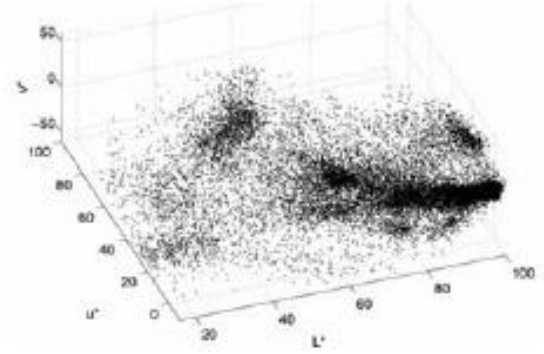


Peak found

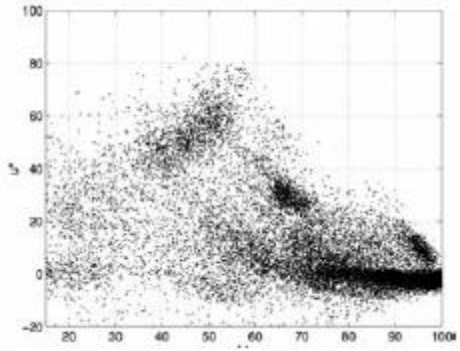


The blue data points were traversed by the windows towards the mode

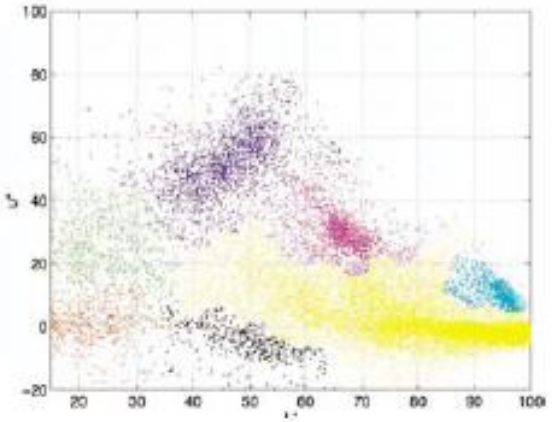
a) input color image



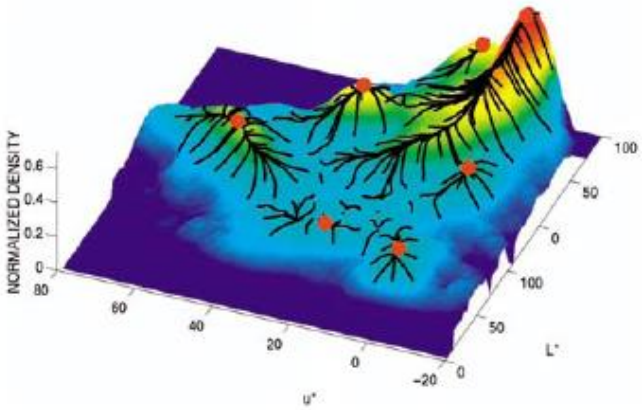
b) pixels plotted in L*u*v* space



c) L*u* space distribution



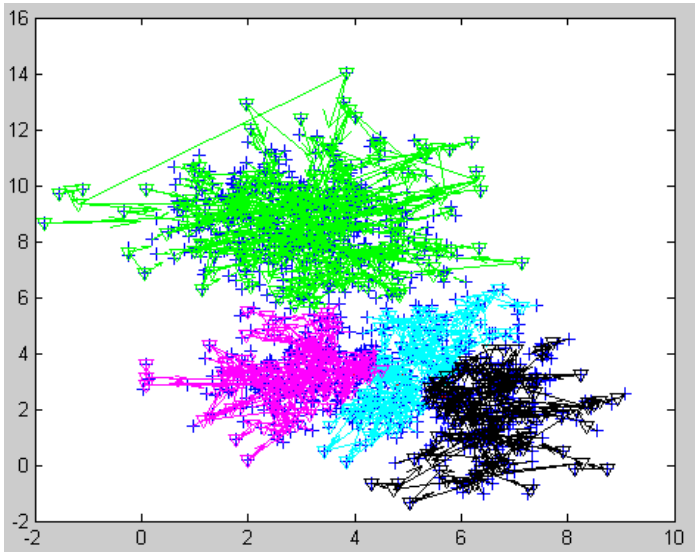
d) clustered results
after 159 mean-shift procedures;



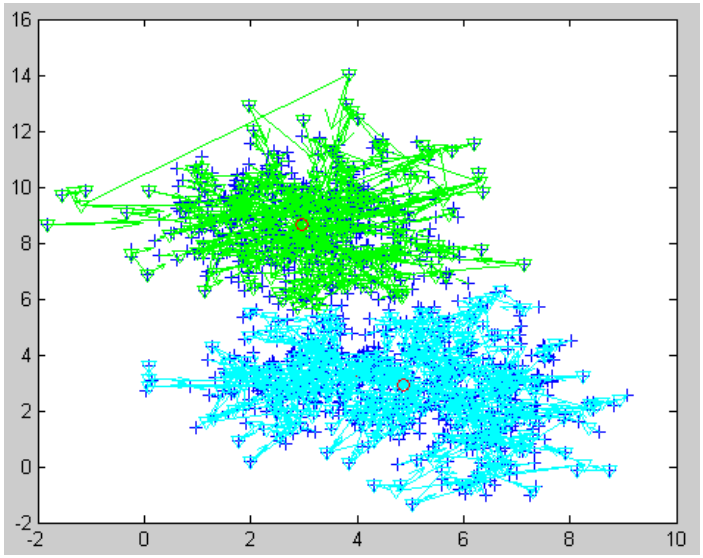
e) corresponding trajectories
with peaks marked as red dots

Mean-shift image segmentation (Comaniciu and Meer 2002).

- Mean-shift is scale (search window size) sensitive.
  - Solution, use several scales.



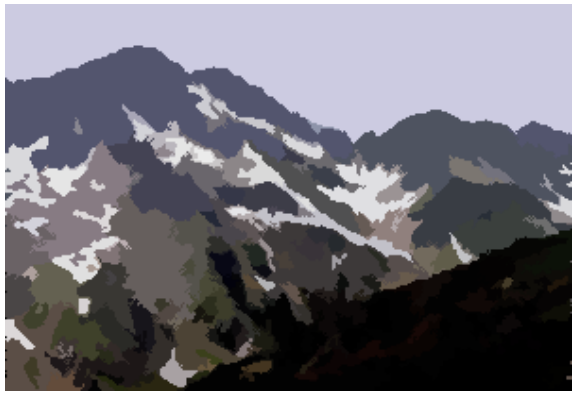
Best 4 clusters



Best 2 clusters

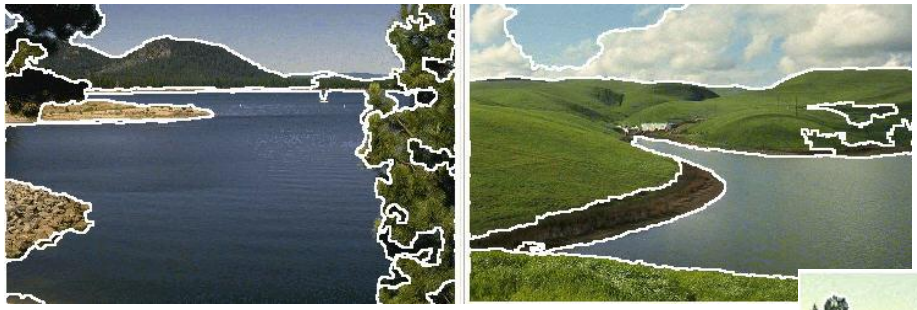
Mean-shift segmentation: results

Mean-shift
color image segmentation,
Comaniciu and Meer 2002

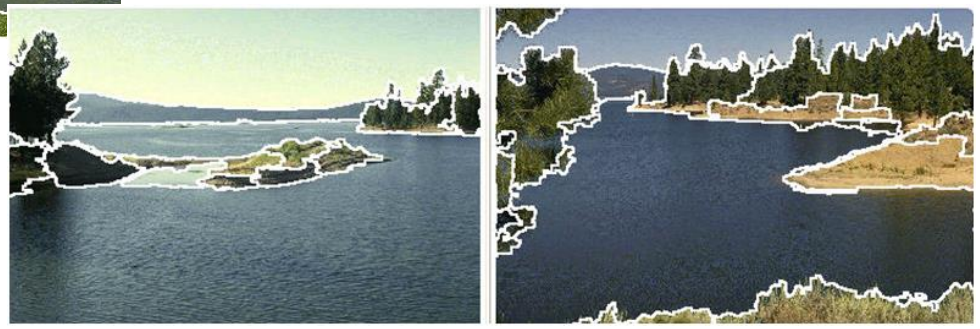


http://www.caip.rutgers.edu/~comaniciu/MSPAMI/msPamiResults.html

Mean-shift segmentation: results



Besides segmentation
meanshift act like
an edge preserving smoothing



Mean-shift pros & cons

- Pros
  - Does not assume spherical clusters
  - Just a single parameter (window size)
  - Finds variable number of modes
  - Robust to outliers
- Cons
  - Computationally expensive
  - Does not scale well with dimension of feature space
  - Output depends on window size

Camshift

- **Camshift (Continuously adaptive mean-shift)**
  is a tracking algorithm related with mean-shift
- If differs from mean-shift in that
  the search window adjusts itself in size.
- If you have well-segmented distributions
  (e.g. face features that stay compact),
  then this algorithm will automatically adjust itself
  for the size of face
  as the person moves closer to and further from the camera.

- https://opencv-python-tutroals.readthedocs.org/en/latest/py_tutorials/py_video/py_meanshift/py_meanshift.html
- https://www.youtube.com/watch?v=RG5uV_h50b0

- **cv::pyrMeanShiftFiltering**
  - Performs initial step of mean-shift segmentation of an image *(the previously described mean-shift algorithm)*
  - see *meanshift_segmentation.cpp f*or an example

- **cv::meanShift**
  - **Finds an object on a back projection image (see next slides)**.
  - The function implements the iterative object search algorithm.
  - It takes the input back projection (*see calcBackProject*) of an object and the initial position.
  - The mass center in window of the back projection image is computed and the search window center shifts to the mass center.
  - The procedure is repeated until the specified number of iterations criteria.maxCount is done or until the window center shifts by less than criteria.epsilon .
  - The algorithm is used inside CamShift() and, unlike CamShift() , the search window size or orientation do not change during the search.
  - You can simply pass the output of calcBackProject() to this function. But better results can be obtained if you pre-filter the back projection and remove the noise. For example, you can do this by retrieving connected components with findContours() , throwing away contours with small area (contourArea()), and rendering the remaining contours with drawContours() .

- **cv::CamShift**
  - Finds an object center, size, and orientation.
    - The function implements the CAMSHIFT object tracking algrorithm [Bradski98].
    - First, it finds an object center using meanShift() and then adjusts the window size and finds the optimal rotation.
    - Returns the rotated rectangle structure that includes the object position, size, and orientation.
  - See *camshiftdemo.cpp* that tracks colored objects

https://github.com/opencv/opencv_attic/blob/master/opencv/samples/cpp/meanshift_segmentation.cpp
https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_video/py_meanshift/py_meanshift.html

- **int meanShift(InputArray probImage, Rect& window, TermCriteria criteria)**

- Finds an object on a back projection image.

- Parameters:
- probImage – Back projection of the object histogram; see calcBackProject() for details.
- window – Initial search window.
- criteria – Stop criteria for the iterative search algorithm.

- The function implements the iterative object search algorithm.
- It takes the input back projection of an object and the initial position.
- The mass center in window of the back projection image is computed and the search window center shifts to the mass center.
- The procedure is repeated until the specified number of iterations criteria.maxCount is done or until the window center shifts by less than criteria.epsilon .
- The algorithm is used inside CamShift() and, unlike CamShift() , the search window size or orientation do not change during the search.
- You can simply pass the output of calcBackProject() to this function. But better results can be obtained if you pre-filter the back projection and remove the noise. For example, you can do this by retrieving connected components with findContours() , throwing away contours with small area (contourArea()), and rendering the remaining contours with drawContours() .

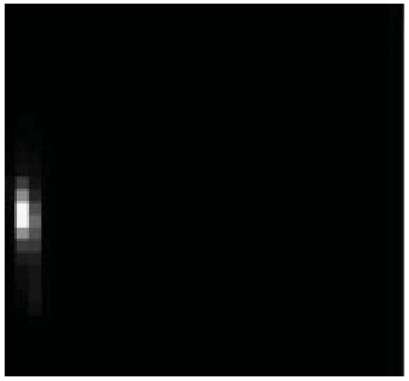
Histogram back projection

- What is Back Projection?
  - Back Projection is a way of recording how well the pixels of a given image fit the distribution of pixels in a histogram model.
  - To make it simpler: for Back Projection, you calculate the histogram model of a feature and then use it to find this feature in an image.
  - Application example: If you have a histogram of skin color (say, a Hue-Saturation histogram), then you can use it to find skin color areas in an image.

Histogram backprojection

- How does it work?
  - Let's say you have gotten a skin histogram (Hue-Saturation) based on the image on the left. The histogram is going to be our model histogram (which we know represents a sample of skin tonality).
    - You applied some mask to capture only the histogram of the skin area.
  - What we want to do is to use our model histogram (that we know represents a skin tonality) to detect skin areas in our test image.
  - In terms of statistics, the values stored in backprojection represent the probability (=> *normalize max. to 1*) that a pixel in test image belongs to a skin area, based on the model histogram that we use.

Hand ("model" of tonality that one wants to detect)



Hue-Saturation histogram of the hand

Test image

BackProjection image

IMAGE SEGMENTATION

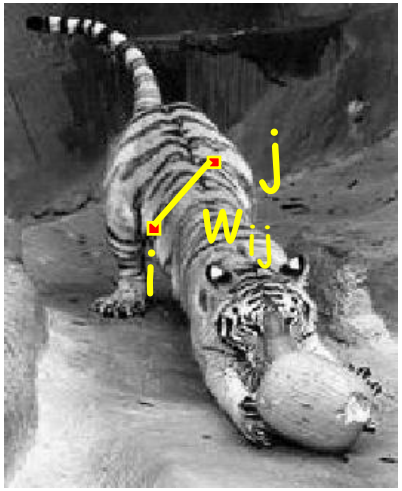
Graph cuts

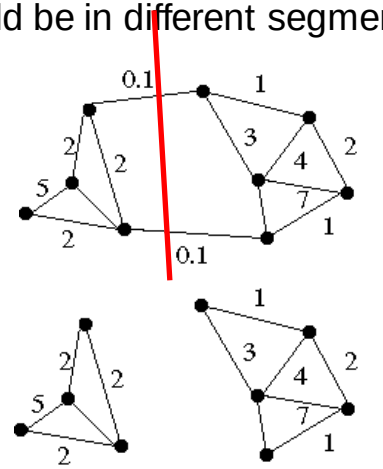
- Represent image as a graph
  - A vertex (node) for each pixel (!)
  - Edges between pixels (!!)
- Weights on edges reflect similarity (affinity) in:
  - Brightness
  - Color
  - Texture
  - Distance
  - …
- Connectivity:
  - Fully connected: edges between every pair of pixels
  - Partially connected: edges between neighboring pixels

Introduction by Prof. Guillermo Sapiro (Duke University)
http://www.youtube.com/watch?v=HMGX8HXskKk

Images as graphs – graph cuts

- Node for every pixel
- Edge between every pair of pixels (!!!) (or every pair of "sufficiently close" pixels)
- Each edge is weighted by the *affinity* or similarity of the two nodes
- Break graph into segments
  - Delete links that cross between segments
  - Easiest to break links that have low affinity
    - similar pixels should be in the same segments
    - dissimilar pixels should be in different segments



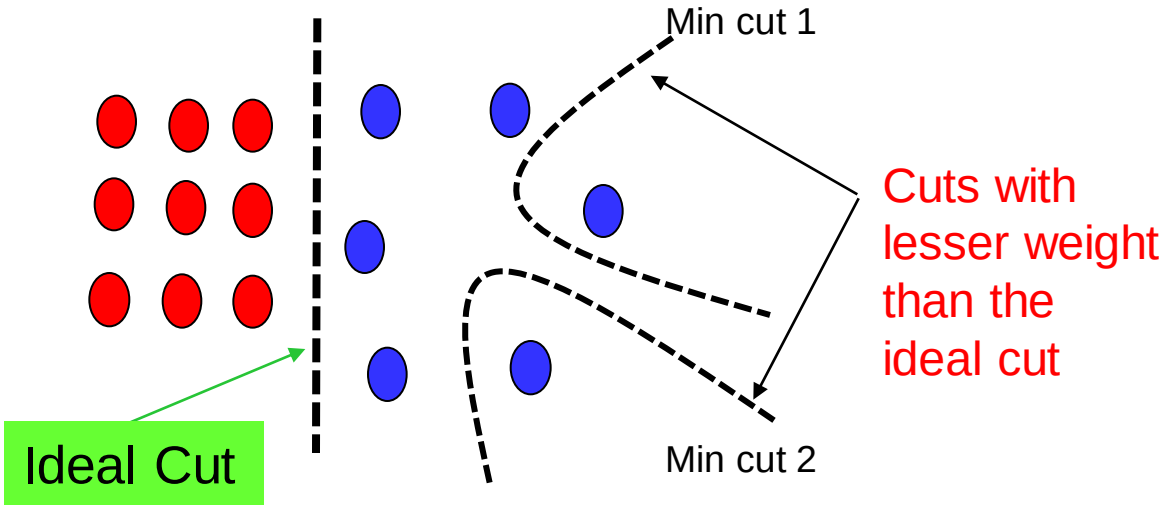


A **cut** of a graph G is the set of edges S such that removal of S from G disconnects G.

Minimum cut is the cut of minimum weight

Minimum cuts vs. Normalized cuts

- There can be more than one minimum cut in a given graph.
- Min cut is not always the best cut
- Using a **minimum cut** as a segmentation criterion,
  does not usually result in reasonable clusters,
  since the smallest cuts usually involve isolating a single pixel.



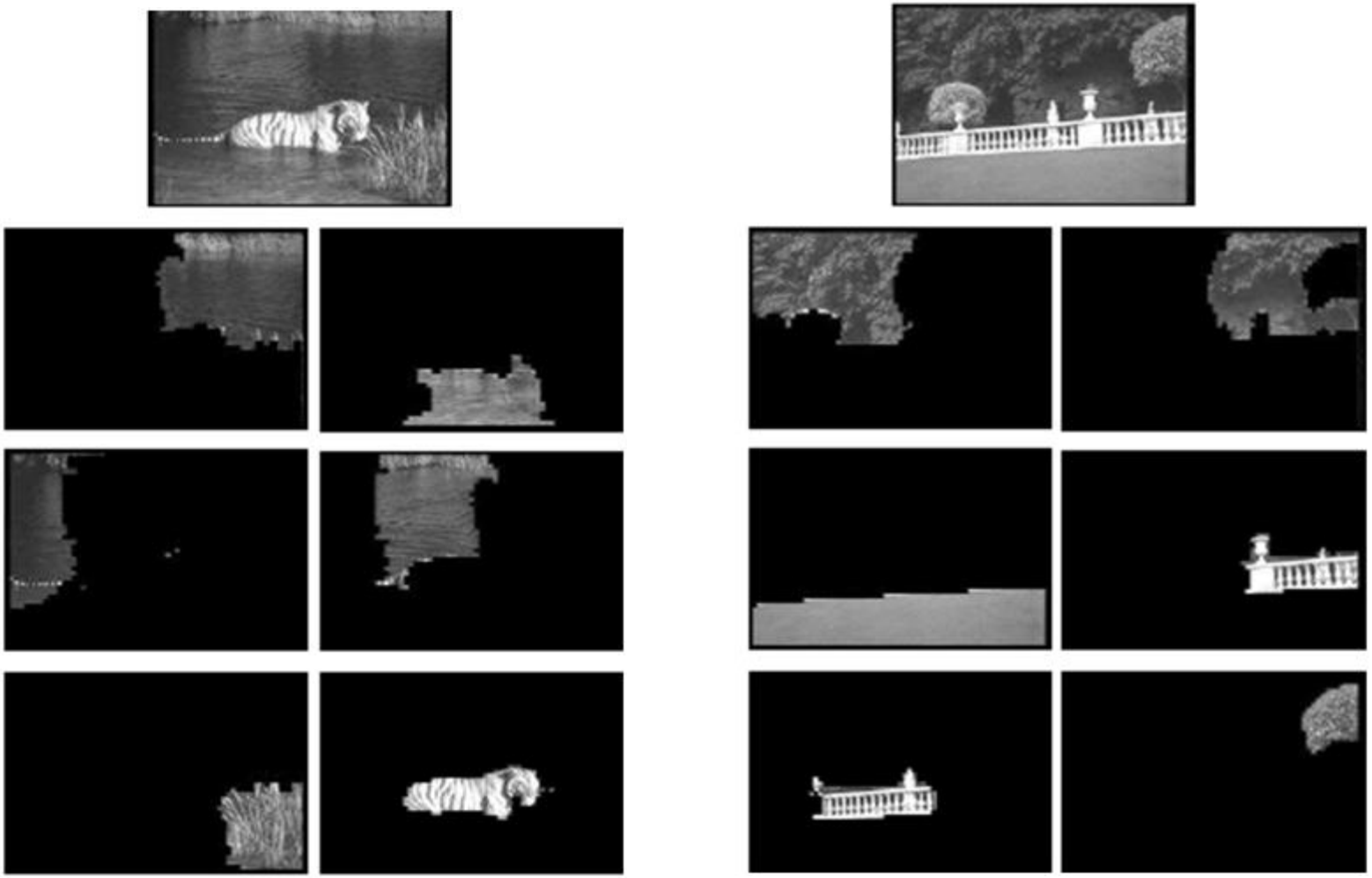
SOLUTION:
normalized cuts

J. Shi and J. Malik, Normalized Cuts and Image Segmentation,
Proc. IEEE Conf. Computer Vision and Pattern Recognition, pp. 731-737, 1997.

Graph cuts

- Pros
  - Generic framework, can be used with many different features and affinity formulations
- Cons
  - Huge storage requirement and time complexity
  - Problems with textured backgrounds
  - Bias towards partitioning into equal segments (???)

Normalized cuts: results



Graph cuts - variants

- **User-interaction:**
  setting seed points or regions (belonging to **object** or **background**) which will be labeled as terminals
- **n-links**: interconnect neighboring pixels
- **t-links**: connect pixels to terminals (object and background)
- A **minimum cost cut** may correspond to segmentation with desirable balance of boundary and regional properties

- *Boykov, Y. and Jolly, M.-P. (2001).*
  Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images.
  In Eighth International Conference on Computer Vision (ICCV 2001), pp. 105–112, Vancouver, Canada.

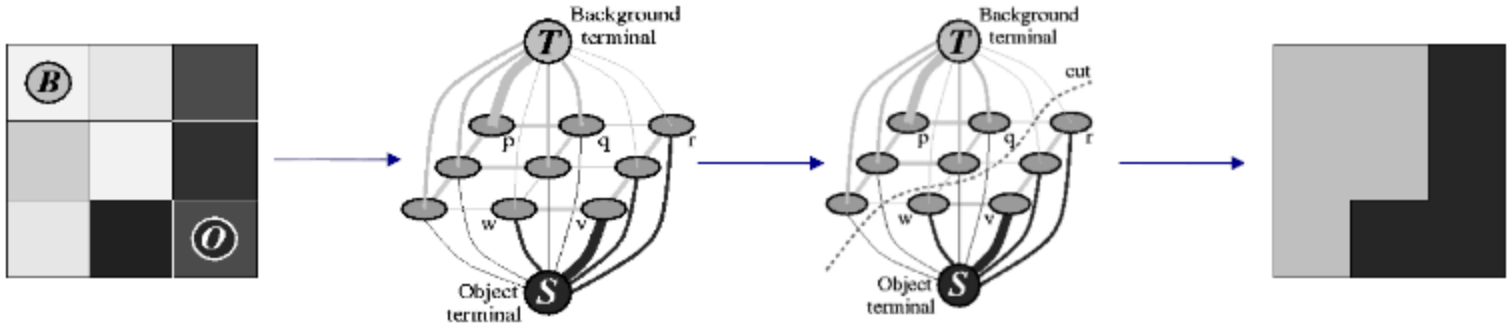
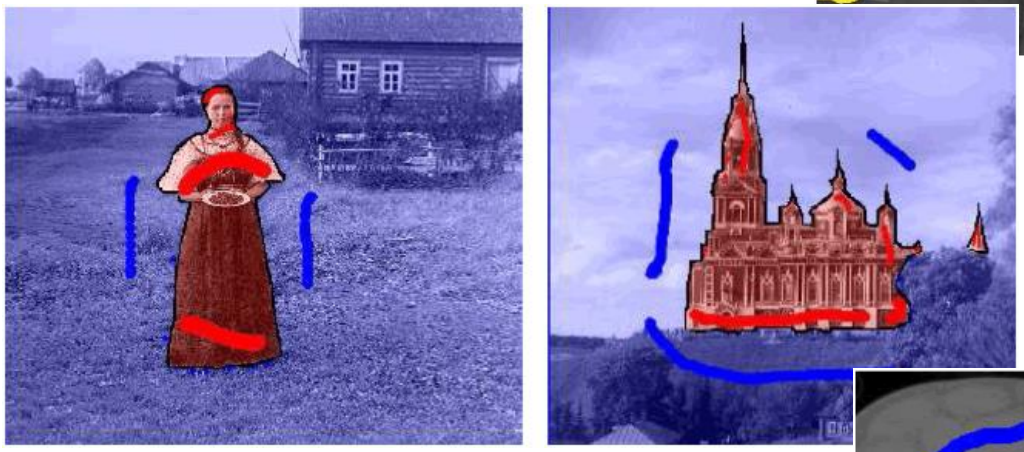


Image with seeds **Graph** **Cut** **Final segmentation**

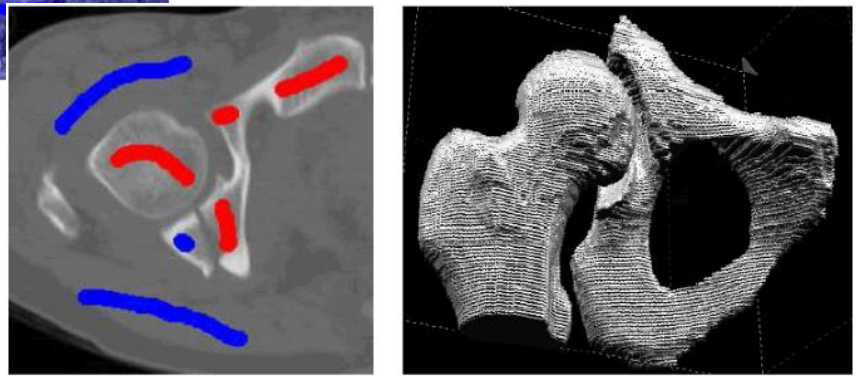
Graph cuts - variants

- RESULTS

Yuri Boykov, Marie-Pierre Jolly: Interactive Graph Cuts for Optimal Boundary & Region Segmentation of Objects in N-D Images. International Conference on Computer Vision (ICCV), 2001



Yuri Boykov, Gareth Funk-Lea. Optimal Object Extraction via Constrained Graph-Cuts. International Journal of Computer Vision, VISI 666-NP



Boykov and Funka-Lea, 3D volumetric medical image segmentation using graph cuts , 2006
(a) computed tomography (CT) slice with some seeds;
(b) recovered 3D volumetric bone model

Graph cuts – variants: GrabCut

- The basic binary segmentation algorithm of Boykov and Jolly (2001) has been extended in a number of directions.
- The **GrabCut system** of Rother, Kolmogorov, and Blake (2004) iteratively re-estimates the region statistics, which are modeled as a mixture of Gaussians in color space.
- This allows their system to operate given minimal user input, such as a single bounding box *(see next slide)*
  - the background color model is initialized from a strip of pixels around the box outline.
  - The foreground color model is initialized from the interior pixels, but quickly converges to a better estimate of the object.
- The user can also place additional strokes to refine the segmentation as the solution progresses.
- In more recent work, Cui, Yang, Wen et al. (2008) use color and edge models derived from previous segmentations of similar objects to improve the local models used in GrabCut.

Graph cuts – variants: GrabCut

- It is the best algorithm to use when one wants to extract a foreground object in a still image (e.g., to cut and paste an object from one picture to another)

(a)

(b)

(c)

source: Szeliski book

(a) the user draws a bounding box in red;
(b) the algorithm guesses color distributions for the object and background and performs a binary segmentation;
(c) the process is repeated with better region statistics.

cv::grabCut

- Input:
  - an image +
  - + a rectangle where the objects are contained
  - OR
  - +a set of pixels that certainly belong to the background/foreground
  - number of iterations
- Procedure
  - A foreground label (cv::GC_PR_FGD) is tentatively assigned to all unmarked pixels
  - Based on the current classification, the algorithm groups the pixels into clusters of similar colors
  - A background/foreground segmentation is done by introducing boundaries between background and foreground pixels
    - done through an optimization process (GrabCut algorithm) that
    - tries to connect pixels with similar labels and
    - imposes a penalty for placing a boundary in regions of relatively uniform intensity
  - The obtained segmentation produces new labels for the pixels
  - The clustering process is repeated and a new optimal segmentation is found again, and so on
  - *Depending on the complexity of the scene, a good solution can be found in more or less iterations (in easy cases, one iteration can be enough)*
- Output:
  - a labelled image (cv::GC_PR_BGD and cv::GC_PR_FGD labels) from which binary images representing the background and the foreground pixels can be extracted

IMAGE SEGMENTATION

Other segmentation methods

Background subtraction

Atlas-based segmentation

Background subtraction

- If we know what the background looks like, it is easy to segment out new regions
- Applications
  - Tracking cars on a road
  - Surveillance
  - Detecting person in an office
  - Video game interfaces
  - …
- Approach
  - estimate background image
  - subtract from current frame
  - large absolute values are interesting pixels

Background estimation

- Offline average
  - Pixel-wise mean values are computed during training phase
- Adjacent frame difference
  - Each image is subtracted from previous image in sequence
- Moving average (running average)
  - Background model is a weighted sum of previous frames

To be studied later .

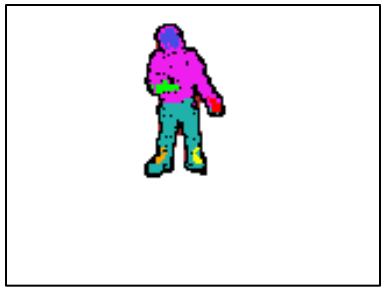
Background subtraction

from C. Stauffer and W. Grimson

Current image

Background image

Foreground pixels



Pfinder, MIT

(recognize human figures and their movements and gestures)

Atlas based segmentation

- Registration of pre-segmented data set (**atlas**) to actual image,
  e.g. by deformable registration
- Mapping of labels from the atlas to the data set
- Segmentation task is converted into registration task

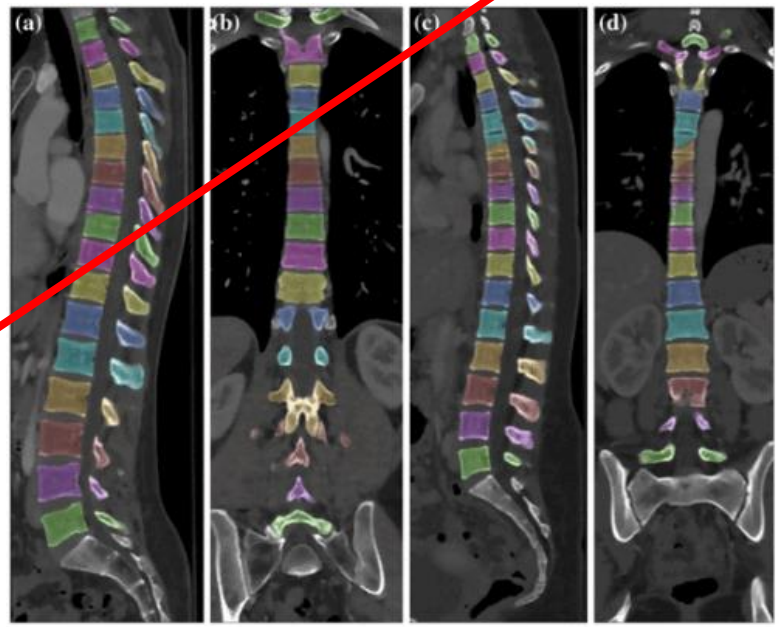


IMAGE PROCESSING & ANALYSIS

Post Processing Steps

Post-processing

- After segmenting an image, we usually want to:
  - Clean up
    the binary image or
    each one of the resulting regions
  - Know something
    about the regions found ...
    - How many objects are in the image?
    - What the the area, perimeter, position, ...
      of the distinct object components ?

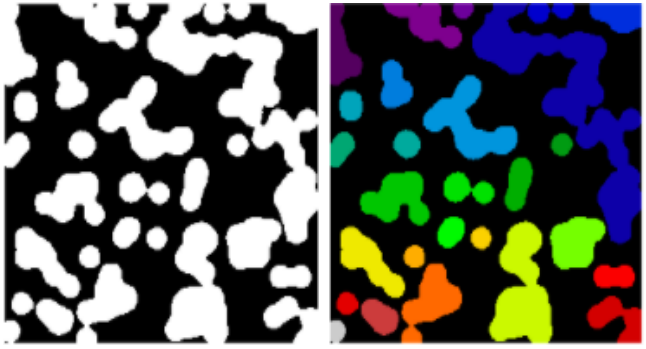
Post-processing

Post-processing steps

- Morphologic operations, for
  - closing gaps and holes
  - removing “noses”
  - computing centerlines
  - ...
- Extraction of connected components (BLOBs)
- Quantitative analysis
  - center, area, permiter, volume, ...
  - principal axes (for orientation)
  - ...
- ...

Extraction of connected components (BLOBs)

- Typically, the segmented image contains multiple objects that should be returned individually.
- Typically, the objects we are interested in are characterized by forming a connected set of pixels.
- To obtain the individual regions we must compute the connected components of the segmented region.
  - => to define when two pixels should be considered connected.
- On a rectangular pixel grid, there are 2 possible definitions of connectivity:
  - 4-connectivity: two pixels are considered part of the same segment if they are horizontally or vertically adjacent
  - 8-connectivity: two pixels are considered part of the same segment if they are horizontally, vertically, or diagonally adjacent
- A **component/region labelling algorithm** determines all the connected components of an image and attributes a unique label to all the points of the same component/region.
- Algorithms:
  - recursive
  - sequential



0	0	0	0	0	0	0
1	0	0	0	1	1	1
1	1	1	0	0	1	0
1	1	0	0	0	1	0
0	0	0	1	0	1	0
0	0	1	1	0	0	0
0	0	0	0	0	1	1

0	0	0	0	0	0	0
1	0	0	0	2	2	2
1	1	1	0	0	2	0
1	1	0	0	0	2	0
0	0	0	3	0	2	0
0	0	3	3	0	0	0
0	0	0	0	0	4	4

http://k-sience.blogspot.com/2017/06/object-counting-using-connected.html
https://scipy-lectures.org/packages/scikit-image/index.html

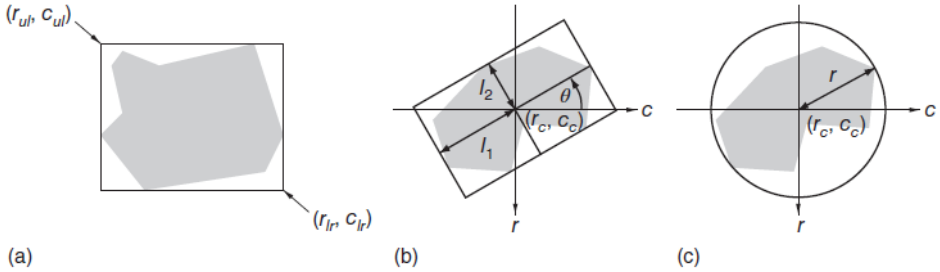
Feature extraction & Shape analysis

- In most applications it is required that we determine one or more characteristic quantities from the segmented regions or contours.
- The quantities we determine are called features; typically they are real numbers.
- The process of determining the features is called feature extraction (*).
- There are different kinds of features:
  - Region and contour features that can be extracted from binary or graylevel images.
- Binary shape analysis:
  - While binary images contain much less information than their grayscale counterparts, they embody shape and size information that is highly relevant for object recognition.
  - Features: area, perimeter, circularity, aspect ratio, moments, … (*see next slide*)
- Gray/Color value features:
  - Use the gray/color values in the image within the region.
  - Features: min, max, mean, standard deviation, histogram(s), moments, …

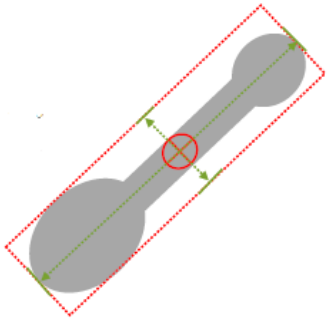
(*) - Do not confuse with "feature point extraction/detection"
(= detection of points that can be differentiated from its neighboring image points)

Feature extraction & Shape analysis

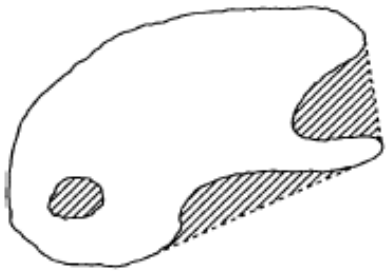
- There are many simple tests of shape that can be made to confirm the identity of objects or to check for items such as defects.
- These include measurements of
  - area and perimeter
  - bounding box, length, width, angle
  - circularity, elongation, aspect ratio
  - length of maximum dimension
  - moments relative to the centroid
    - note: some of the statistics above can be obtained from centralized moments
  - number and area of holes
  - area and dimensions of the convex hull and enclosing rectangle
  - number of sharp corners
  - numbers and types of skeleton nodes
  - ...



(a) Smallest axis-parallel enclosing rectangle of a region.
(b) Smallest enclosing rectangle of arbitrary orientation.
(c) Smallest enclosing circle.



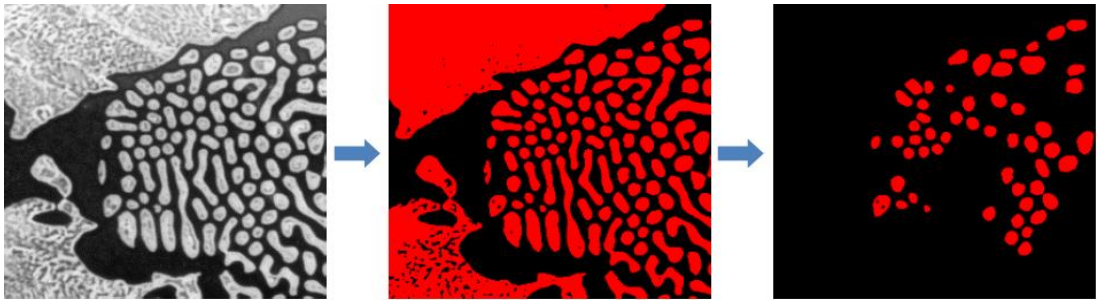
Enclosing rectangle, centroid and main axis



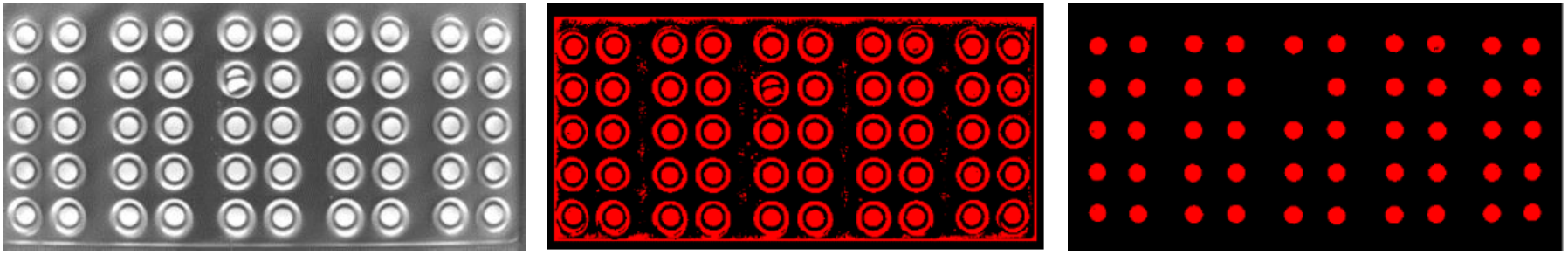
Convex hull and convex deficiency.

Feature extraction: Particle filtering

- Keeps or removes particles based on geometric features
  - area, width, height, aspect ratio and other features are commonly used to filter
- Typically used on binary images
- Cleans up noisy images



Particles with circularity factor < 1.1



Original | Threshold | Particle filter

Structural Analysis and Shape Descriptors

- **cv::connectedComponents**
  - computes the connected components labeled image of boolean image
- **cv::connectedComponentsWithStats**
  - computes the connected components labeled image of boolean image and also produces a statistics output for each label
- **cv::moments**
  - Calculates all of the moments up to the third order of a polygon or rasterized shape
- **cv::HuMoments**
  - Calculates seven Hu invariants
- **cv::findContours**
  - Finds contours in a binary image
- **cv::arcLength**
  - Calculates a contour perimeter or a curve length
- **cv::contourArea**
  - Calculates a contour area
- **cv::boundingRect**
  - Calculates the up-right bounding rectangle of a point set or non-zero pixels of gray-scale image
- **cv::minAreaRect**
  - Finds a rotated rectangle of the minimum area enclosing the input 2D point set
- **cv::minEnclosingCircle**
  - Finds a circle of the minimum area enclosing a 2D point set
- **cv::isContourConvex**
  - Tests a contour convexity
- **cv::convexHull**
  - Finds the convex hull of a point set
- … (and several other)

IMAGE PROCESSING & ANALYSIS

Morphological image processing

Morphological operations

- Segmentation results often contain unwanted noisy parts.
  Furthermore, sometimes the segmentation will contain parts in which the shape of the object we are interested in has been disturbed, for example, because of noise.
- Therefore, often we need to modify the shape of the segmented regions to obtain the desired results.
- This is the subject of the field of **mathematical morphology**, which can be defined as a theory for the analysis of spatial structures.
- Mathematical morphology provides a set of extremely useful operations that enable us not only to modify the shape of objects but also to smooth images or enhance edges (*see next slides*)
- **Morphological image processing**
  - **a set of operations by which the spatial structure of objects within an image is modified.**

- **Morphological operations** can be defined on **binary** and **gray value images**.

- **Operations (binary / grayscale):**
  - Dilation
  - Erosion
  - Opening = Dilation(Erosion)
  - Closing = Erosion(Dilation)

Operations (grayscale):

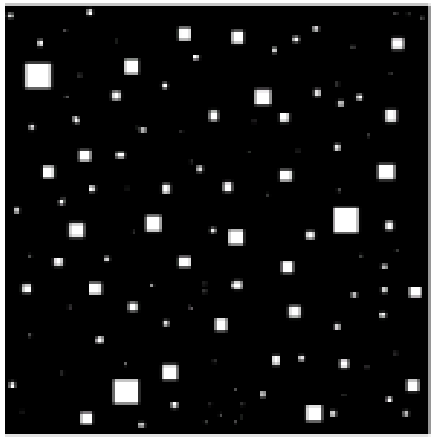
- Morphological gradient
- TopHat - reveals areas that are lighter than the surrounding region
- BottomHat - reveals areas that are darker than the surrounding region

- Two fundamental parameters of those operations are the **shape** and **center** of the **structuring element**.

Morphological operations

Original image

Image after
morphological processing



Original image

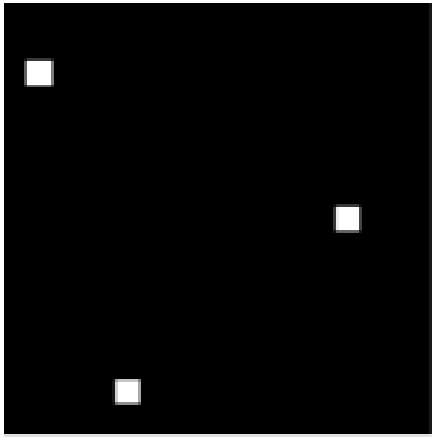
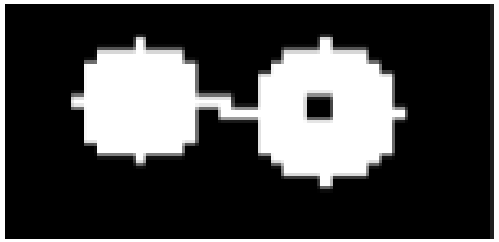


Image after
morphological processing



Original image

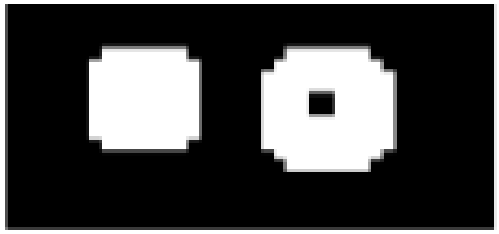


Image after
morphological processing
(opening)

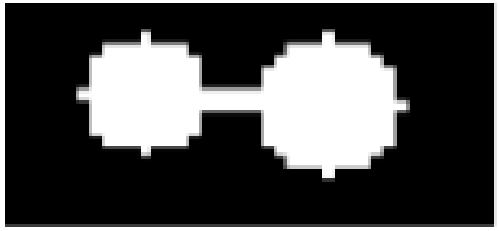
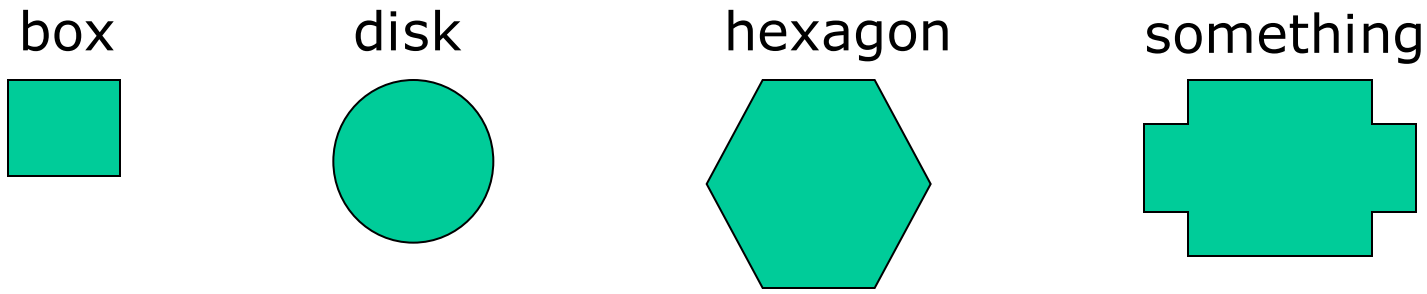


Image after
morphological processing
(closing)

Morphological operations

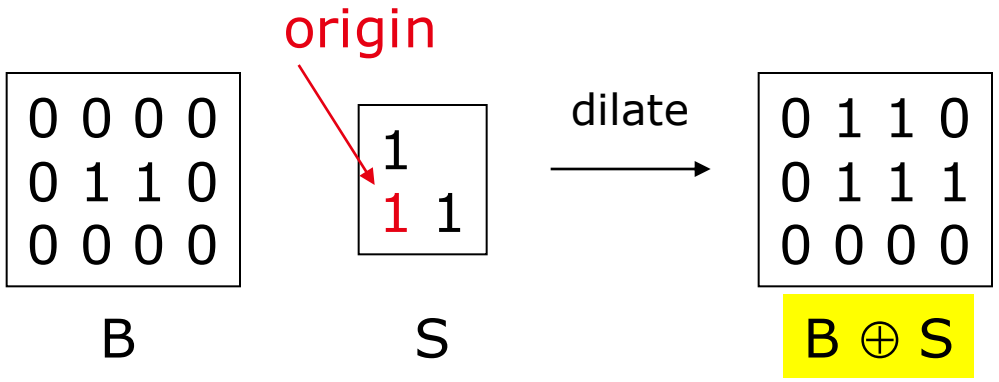
- The arguments to dilation and erosion are:
  - a binary image **B**
  - a structuring element **S**
- **Structuring element (**or **kernel)**
  - A structuring element is a shape mask used in the basic morphological operations.
  - They can be any shape and size that is digitally representable, and each has an origin.
  - It can even correspond to a particular shape that is being sought for in the image.



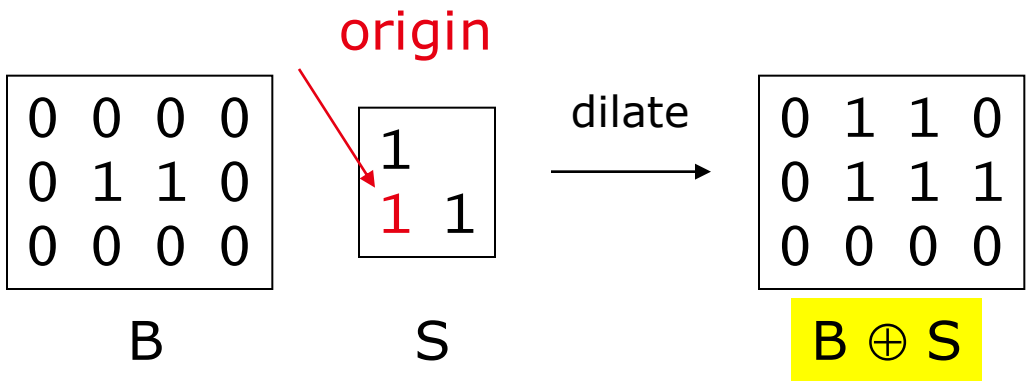
dilate(B,S)

- takes binary image B,
- places the origin of structuring element S over each 1-pixel, and
- ORs the structuring element S into the output image at the corresponding position.

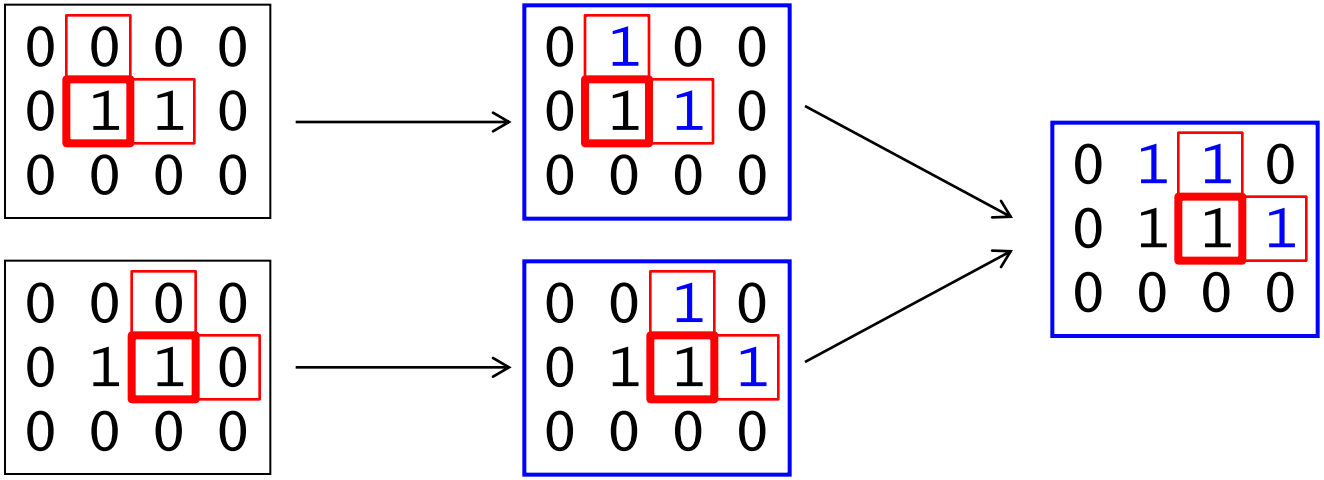
- Dilation is both associative and commutative
  - this fact allows breaking a complex shape into several simpler shapes which can be combined as a sequence of dilations



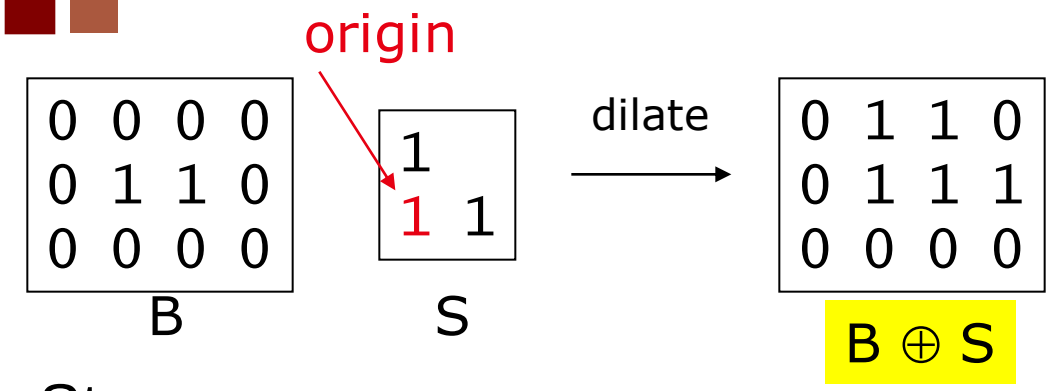
Dilation

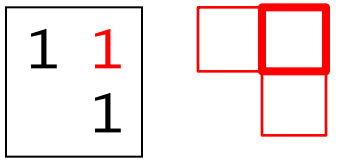


Steps:

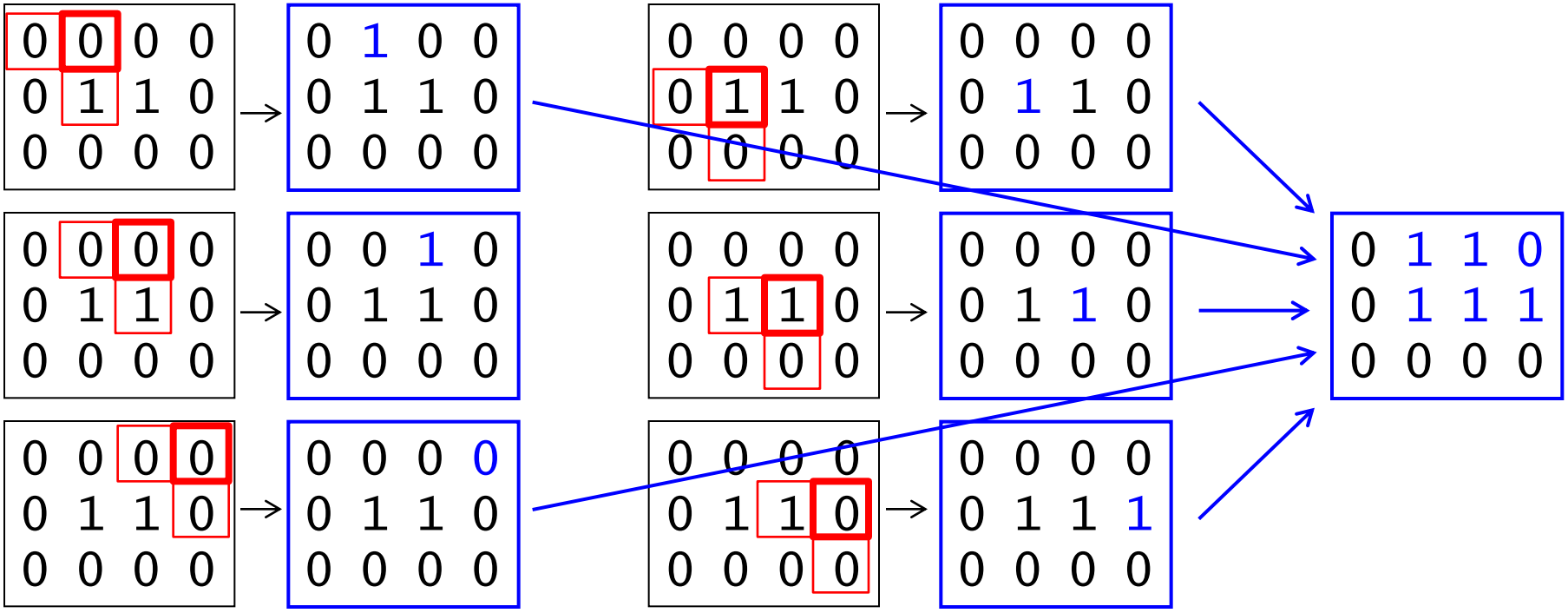


Dilation





Steps (alternative formulation): **dilate (B,S) (x,y) = max {B(x-x´,y-y') | (x´,y´) ∈ S }**

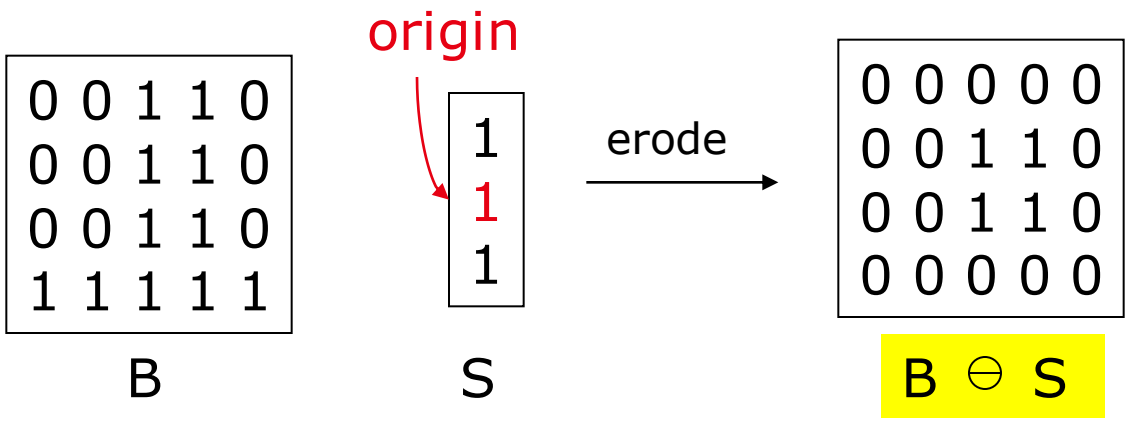


Erosion

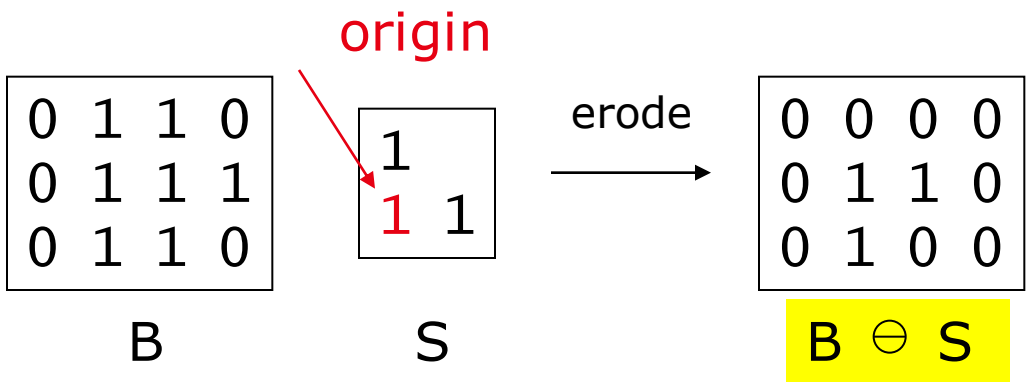
erode(B,S)

- takes a binary image B,
- places the origin of structuring element S over every pixel position, and
- ORs a binary 1 into that position of the output image only if every position of S (with a 1) covers a 1 in B.

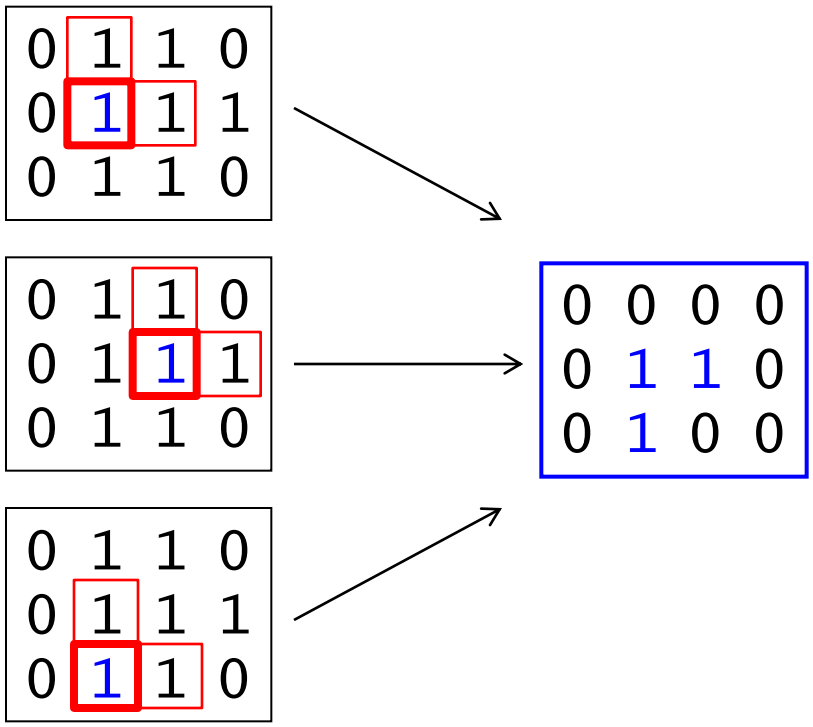
- Note:
  erosion gives all the locations where the structuring element is contained in the image



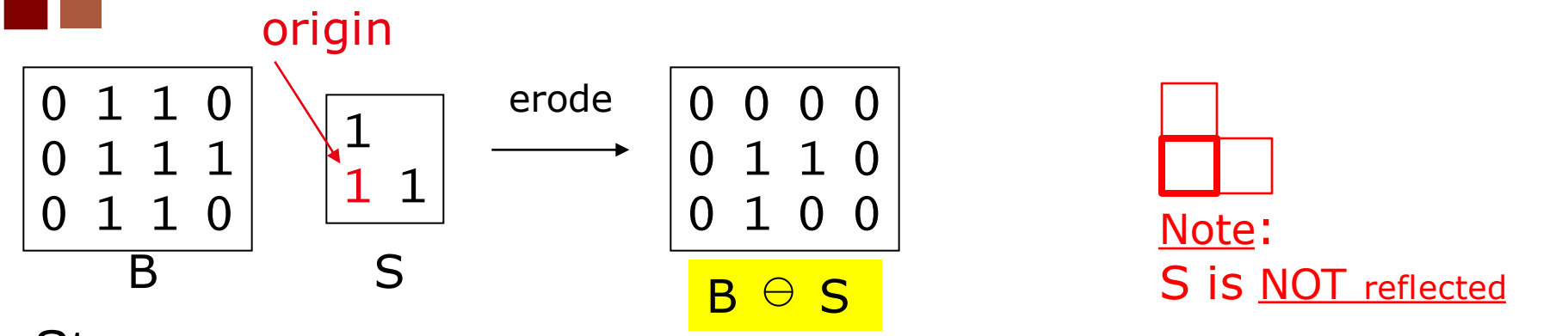
Erosion



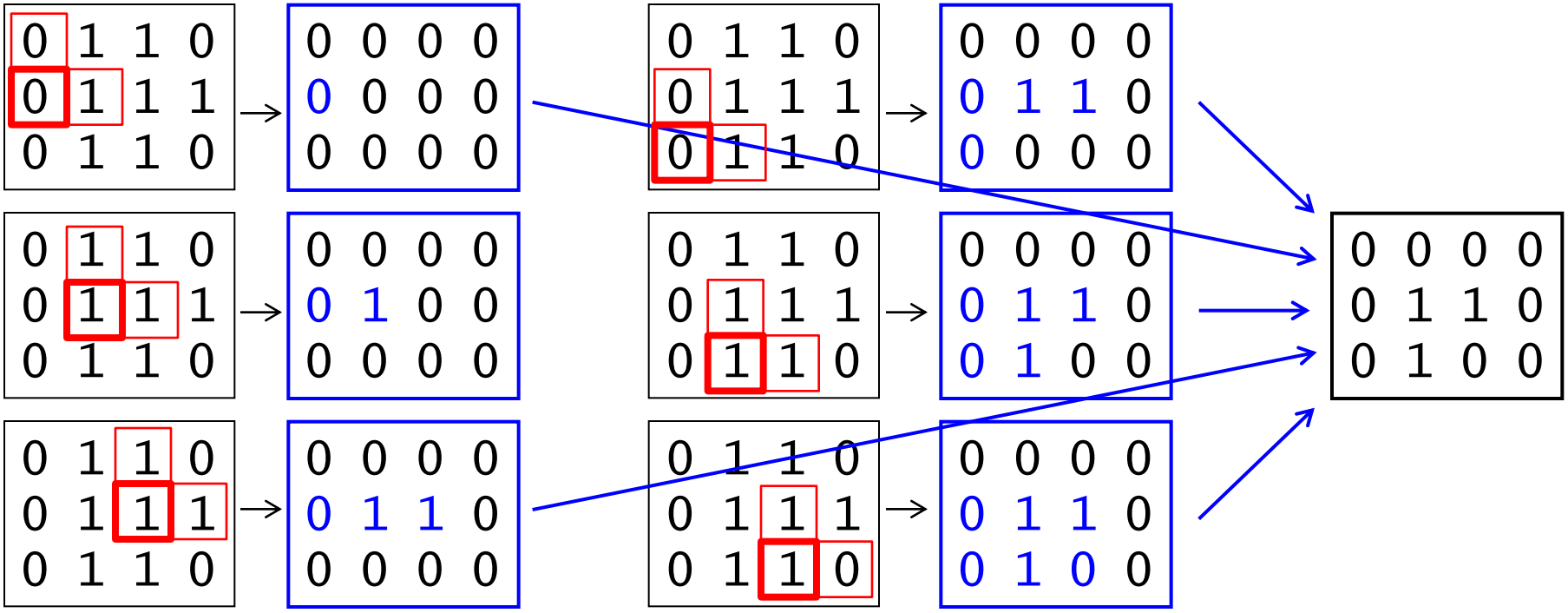
Steps:



Erosion



Steps (alternative formulation): **erode(B,S) (x,y) = min {B(x+x´,y+y') | (x´,y´) ∈ S }**



Dilation and erosion

- Dilation
  - expands the connected sets of 1s of a binary image.
  - it can be used for
    - growing features
    - filling holes and gaps
- Erosion
  - shrinks the connected sets of 1s of a binary image.
  - it can be used for
    - shrinking features
    - removing bridges, branches and small protrusions

- Dilation and erosion can be useful in a wide variety of contexts such as:
  - removing noise,
  - isolating individual elements, and
  - joining disparate elements in an image.

- Morphological operators can also be used
  - to find intensity bumps or holes in an image and
  - to calculate image gradients.

Opening and closing

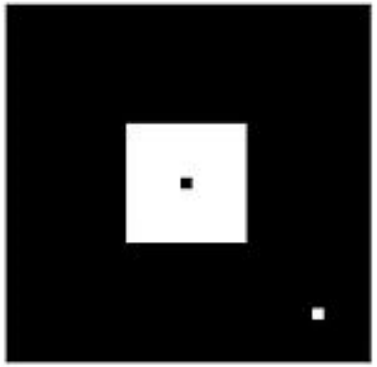
Opening

- is the compound operation of erosion followed by dilation (with the same structuring element)
- **open(B,S)** = dilate(erode(B,S),S)

Closing

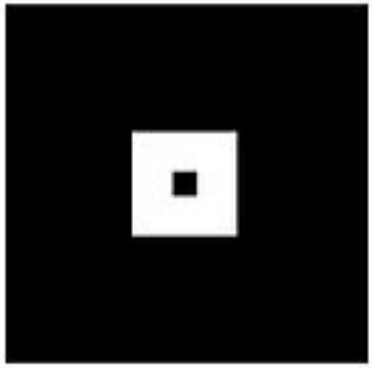
- is the compound operation of dilation followed by erosion (with the same structuring element)
- **close(B,S)** = erode(dilate(B,S),S)

Morphological operations: results



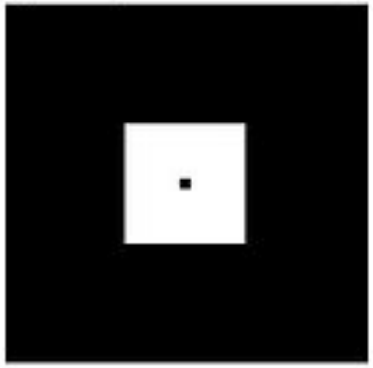
original image

Eroded



Dilated

Opened (Dilation after Erosion)



Closed (Erosion after Dilation)

source:
https://en.wikibooks.org/wiki/Sensory_Systems/Computer_Models/Descriptive_Simulations_of_Visual_Information_Processing

Morphological operations on graylevel

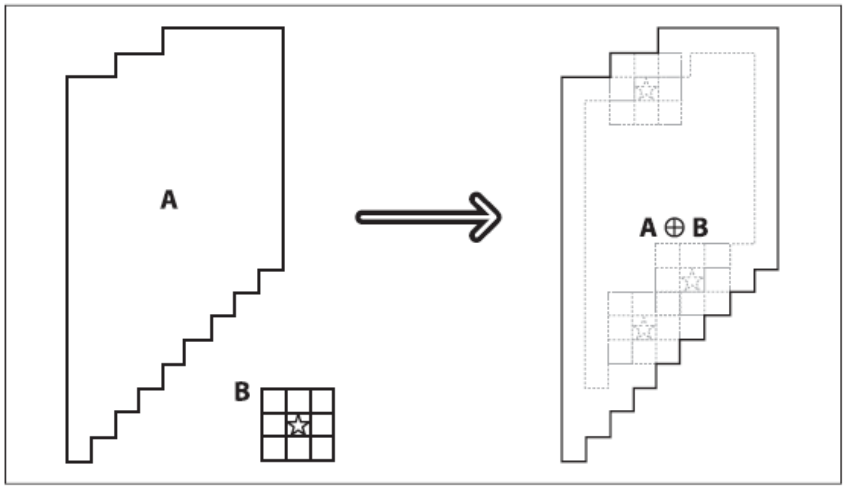
- Dilation
  - $(g \oplus s)(x,y) = \max \{g(x-x',y-y') + s(x',y') | (x',y') \in D_s\}$
- Erosion
  - $(g \ominus s)(x,y) = \min \{g(x+x',y+y') - s(x',y') | (x',y') \in D_s\}$
- being
  - g(x,y) a graylevel image
  - s(x,y) a structuring element (graylevel)
    - D_s a binary matrix that defines the locations in the neighborhood that must be included in the max/min operations
- When the structuring element is flat one can define $s(x',y') = 0$ for $(x',y') \in D_s$
- So ...
  - Dilation
    - $(g \oplus s)(x,y) = \max \{g(x-x',y-y') | (x',y') \in D_s\}$
  - Erosion
    - $(g \ominus s)(x,y) = \min \{g(x+x',y+y') | (x',y') \in D_s\}$
- ...and if the structuring element is symmetric ...
  - Dilation
    - $(g \oplus s)(x,y) = \max \{g(x+x',y+y') | (x',y') \in D_s\}$
  - Erosion
    - $(g \ominus s)(x,y) = \min \{g(x+x',y+y') | (x',y') \in D_s\}$

Dilation & Erosion

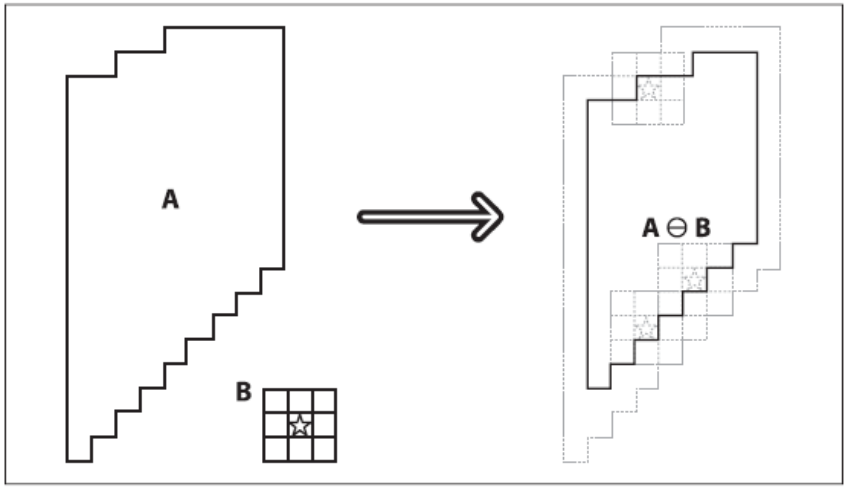
$$\text{erode}(x,y) = \min_{(x',y')\in \text{kernel}} \text{src}(x+x', y+y')$$

$$\text{dilate}(x,y) = \max_{(x',y')\in \text{kernel}} \text{src}(x+x', y+y')$$

kernel ⇔ structuring element



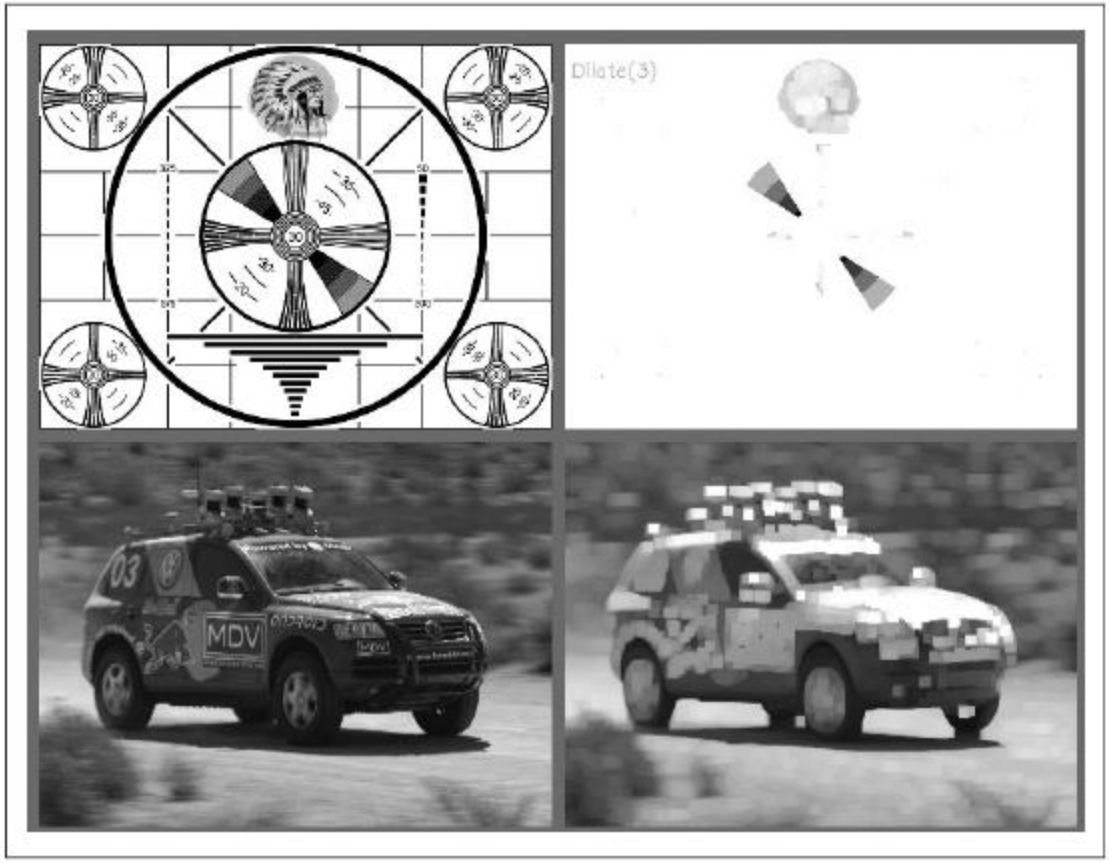
Dilation: take the **maximum** under the kernel B



Erosion: take the **minimum** under the kernel B

(source: Bradski book)

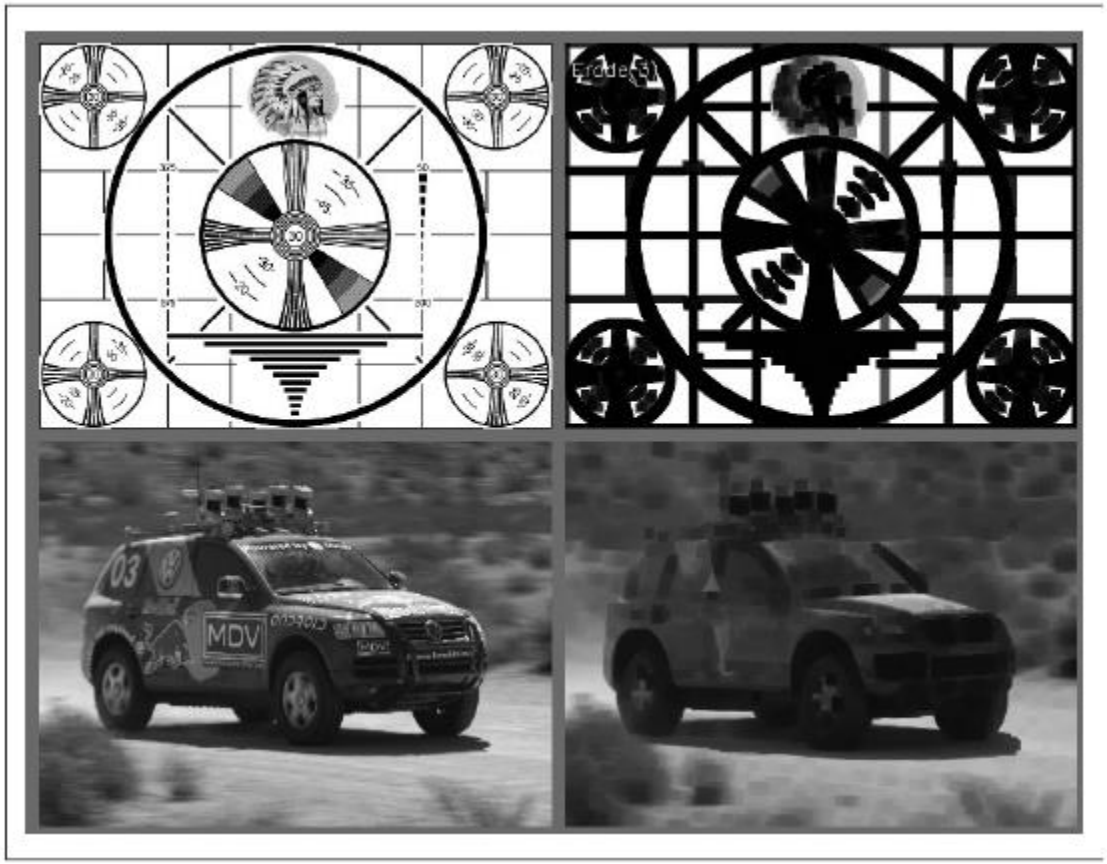
Dilation



Results of the dilation, or “max”, operator:
bright regions are expanded and often joined
(source: Bradski book)

(source: Bradski book)

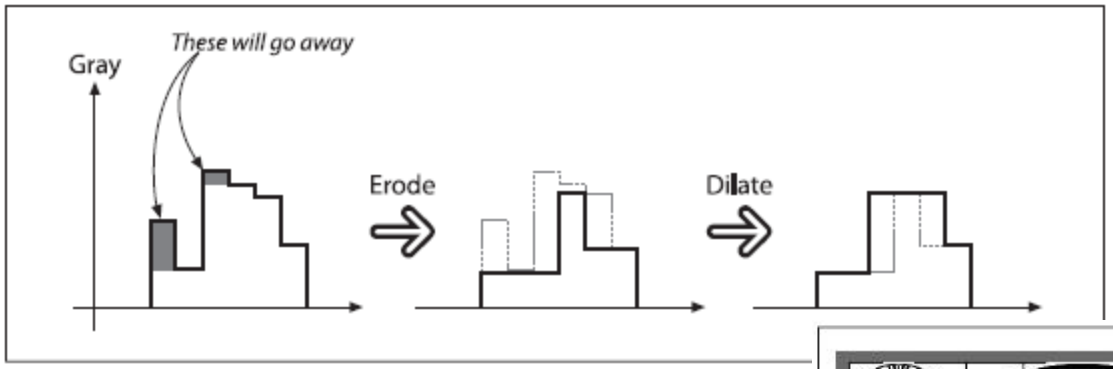
Erosion



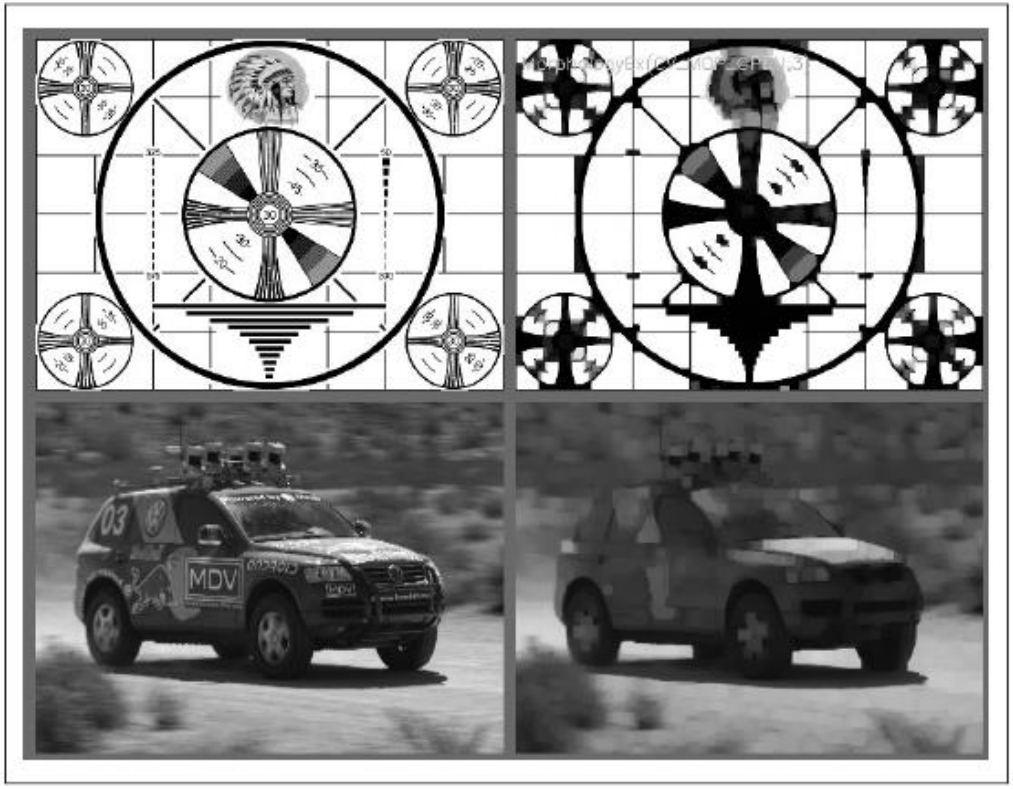
Results of the erosion, or “min”, operator:
bright regions are isolated and shrunk
(source: Bradski book)

(source: Bradski book)

Opening



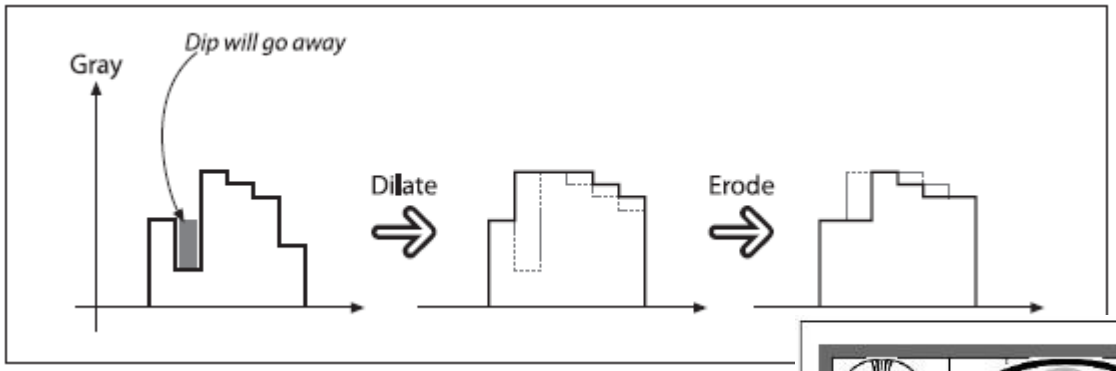
Morphological opening:
the upward outliers
are eliminated as a result



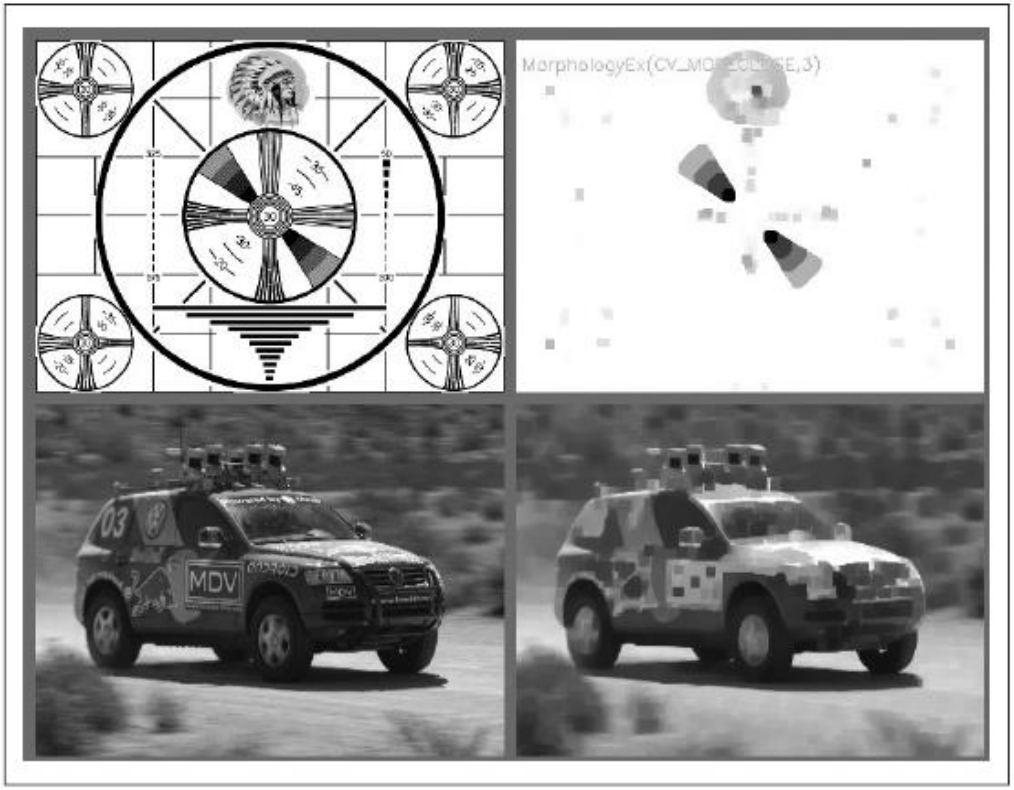
Results of opening on an image:
small bright regions are removed, and
the remaining bright regions
are isolated but retain their size
(source: Bradski book)

(source: Bradski book)

Closing



Morphological closing:
the downward outliers
are eliminated as a result



Results of closing on an image:
bright regions are joined
but retain their basic size
(source: Bradski book)

(source: Bradski book)

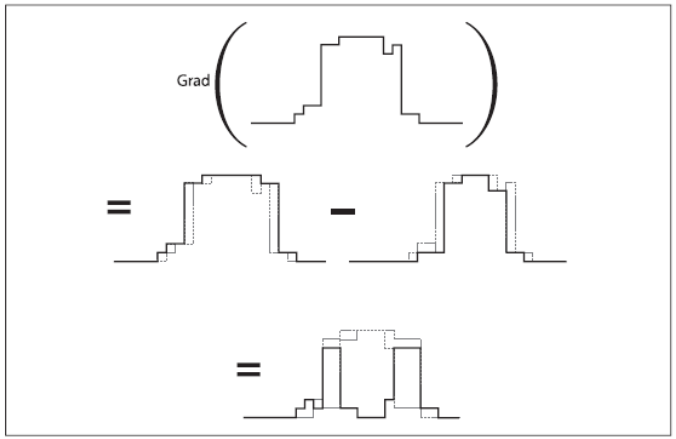
Other morphological operators

- Morphological gradient
  - **gradient(src) = dilate(src)–erode(src)**
    - the effect of this operation on a binary image would be simply to isolate perimeters of existing blobs.
- Top-hat *
  - **TopHat(src) = src–open(src)**
    - reveals areas that are lighter than the surrounding region
- Black-hat **
  - **BlackHat(src) = close(src)–src**
    - reveals areas that are darker than the surrounding region

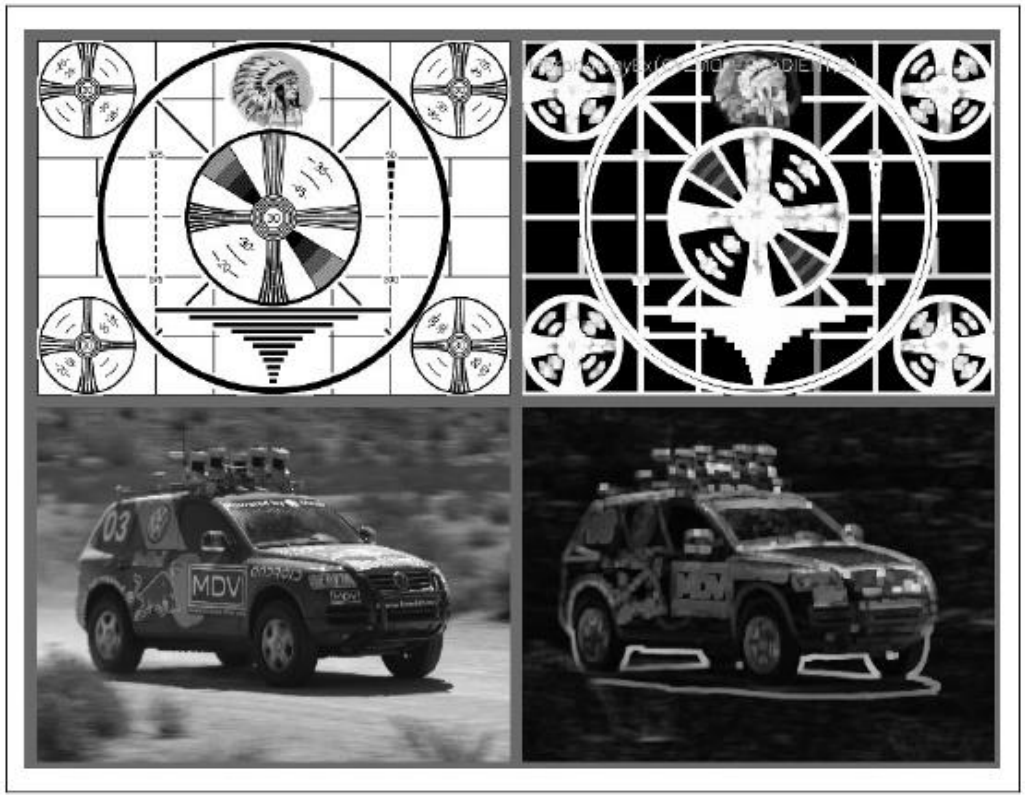
*, ** - Bradski book ("Learning OpenCV") designations;
"Black-hat" is also known as "Bottom-hat"

Morphological gradient

Morphological gradient



Morphological gradient
applied to a grayscale image:
as expected,
the operator has its highest values
where the grayscale image
is changing most rapidly



Results of the morphological gradient operator:
bright perimeter edges are identified

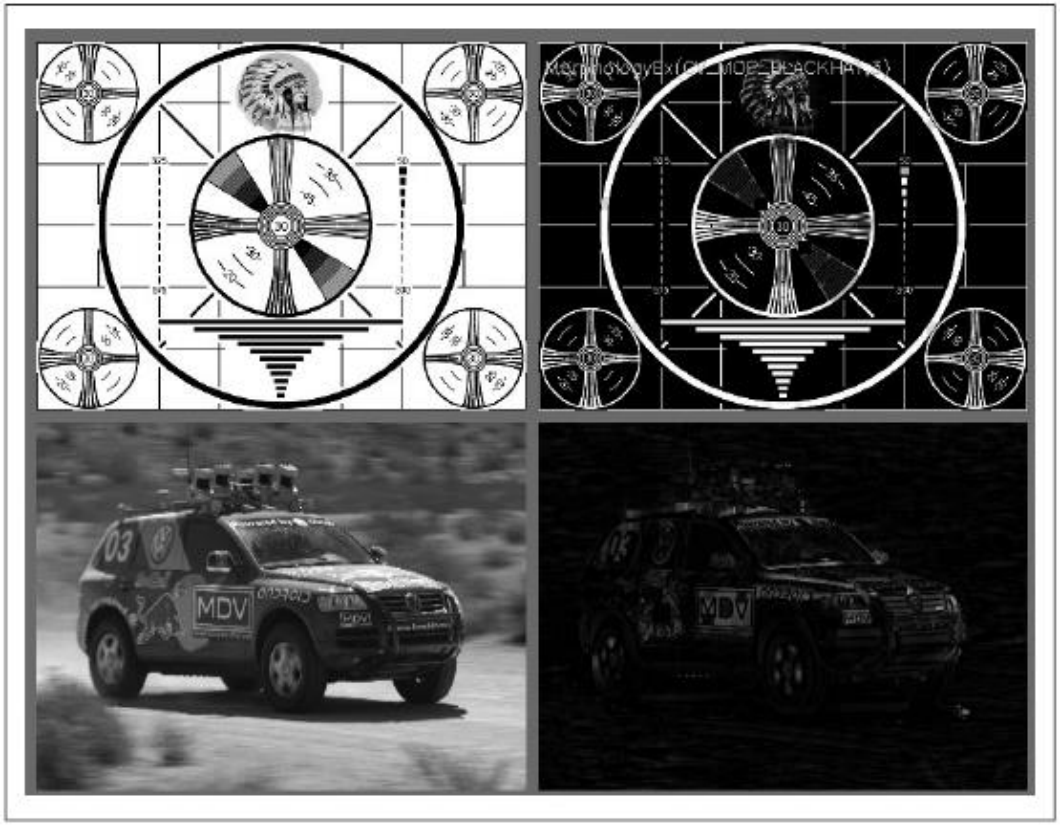
(source: Bradski book)

Top-hat

Results of Top-Hat:
bright local peaks are isolated

(source: Bradski book)

Black-hat



Results of Black-Hat operation:
dark holes are isolated

(source: Bradski book)

Morphological operations examples

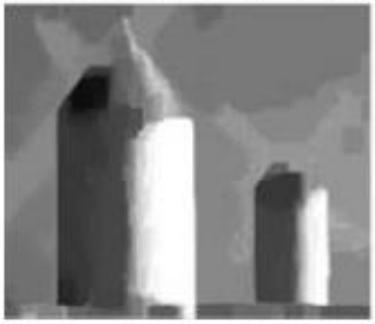
Image I

Erosion I⊖B

Dialation I⊕B

Opening IoB= (I⊖B)⊕B

Closing I•B= (I⊕B)⊖B



Grad(I)= (I⊕B)-(I⊖B)

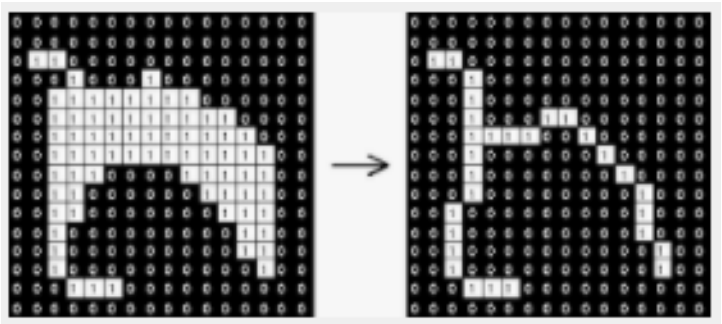
TopHat(I)= I - (I⊖B)

BlackHat(I)= (I⊖B) -1

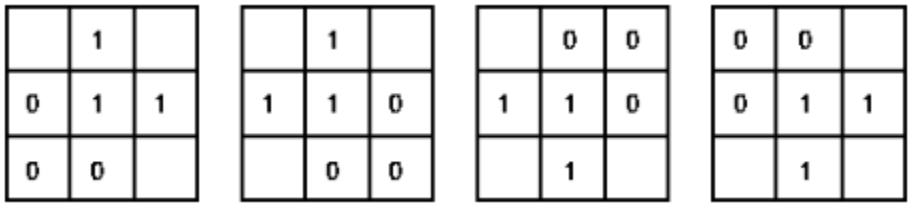
Other morphological operations

- Hit-and-miss transform
  - can be used to search an image for particular instances of a shape or other characteristic image feature
- Thinning
- Thickening
- Skeletonization / Medial Axis Transform

http://homepages.inf.ed.ac.uk/rbf/HIPR2/morops.htm

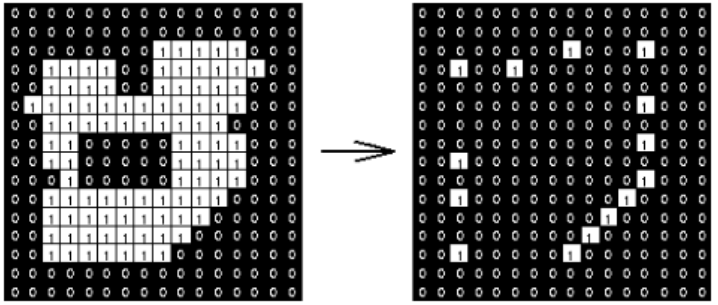


Skeletonization by morphological thinning



Four structuring elements used
for corner finding in binary images
using the **hit-and-miss transform**.
Note that they are really all the same element,
but rotated by different amounts.

https://homepages.inf.ed.ac.uk/rbf/HIPR2/hitmiss.htm



Effect of the hit-and-miss based
90º angle convex corner detector
on a simple binary image

Morphological operators usage

- Opening
  - often used before counting regions in a binary image.
    - Ex: a thresholded image of cells on a microscope slide, to separate out cells that are near each other before counting the regions.
- Closing
  - used in most of the more sophisticated connected-component algorithms to reduce unwanted or noise-driven segments.

- Note:
  - open and close tend to preserve the area of connected regions.

- Morphological gradient
  - often used when we want to isolate the perimeters of bright regions so we can treat them as whole objects (or as whole parts of objects).
  - The complete perimeter of a region tends to be found because an expanded version is subtracted from a contracted version of the region, leaving a complete perimeter edge.

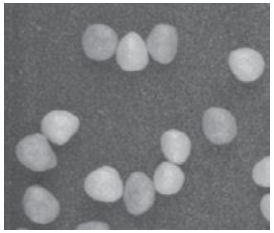
Morphological operators usage

Morphological operators usage:

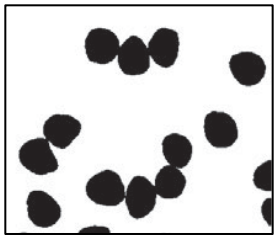
- Binary images
  - Remove unwanted information caused by the segmentation process (ex: thresholding):
    - Noise particles
    - Holes within particles
    - Particles touching the border of an image
    - Particles touching each other
    - Particles with uneven borders
  - Extract the border of objects (morphological gradient)
  - Extract the skeleton of the object (and from it, endpoints and junctions)
    - can be used for recognition purposes
- Grayscale images
  - Smoothing: opening followed by a closing operation; it removes both bright and dark artifacts of noise
  - Edge enhancement (morphological gradient)
  - Reveal are that are lighter (TopHat) or darker (BlackHat*) than the surrounding region
    (*) OpenCV library name
- When used correctly,
  morphological operations preserve the essential features of shape of an object,
  while removing the irrelevant details.

Morphological operators: Binary images

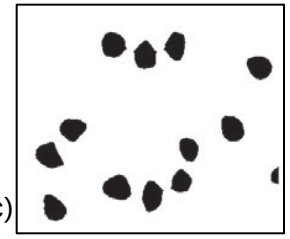
(a)



(b)



(c)



(a) Image of several globular objects.

(b) Result of thresholding (a)

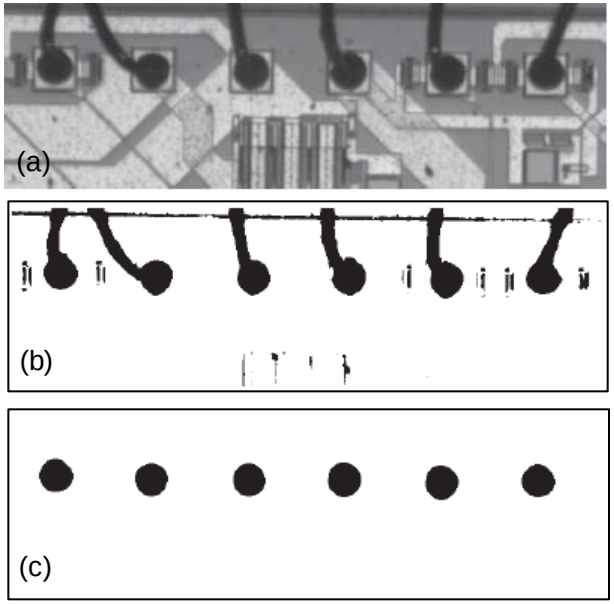
(c) Result of **eroding the inverse of (b)** with a circle of diameter 15

Note: in these 2 examples, the objects are black and the background is white

(a) Image of a ball-bonded die.
The goal is to segment the balls.

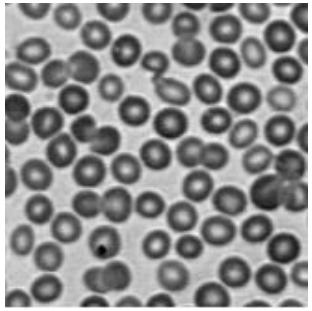
(b) Result of thresholding (a).
The segmentation includes the wires that are bonded to the pads.

(c) Result of performing an **opening of the inverse of (b)** with a circle of diameter 31.
The wires and the other extraneous segmentation results have been removed by the opening, and only the balls remain.

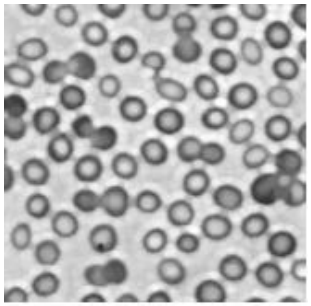


Adapted from Handbook of Machine and Computer Vision
A. Hornberg (editor)

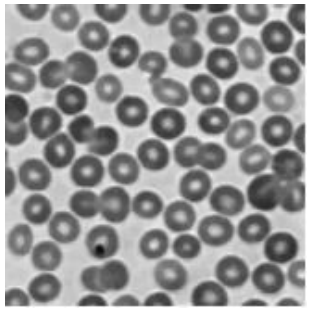
Morphological operators: Gray scale images



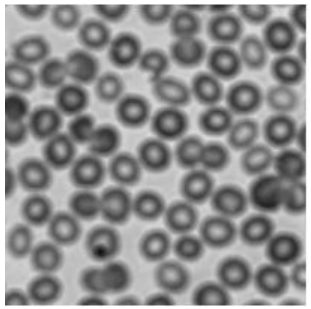
original



dilation

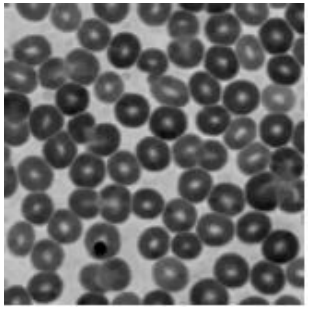


closing

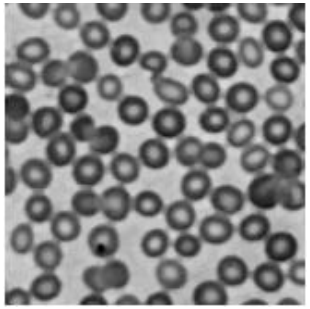


(dilation+erosion)/2
~= smoothing

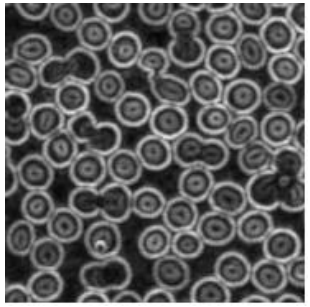
structuring element …?



erosion



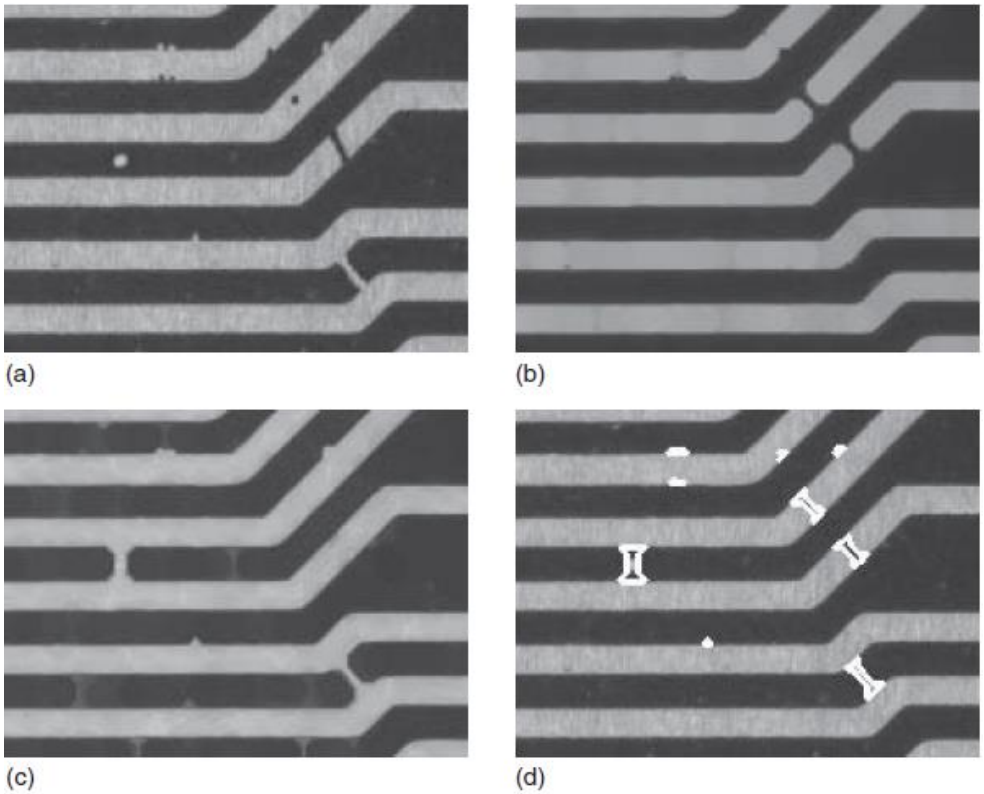
opening



morphologic gradient

http://www.cyto.purdue.edu/cdroms/micro2/content/education/wirth08.pdf

Morphological operators: Gray scale images



(a) Image showing a **part of a PCB** with several tracks that have spurs, mouse bites, pinholes, spurious copper, and open and short circuits.

(b) Result of performing a gray value opening with an octagon of diameter 11 on (a).

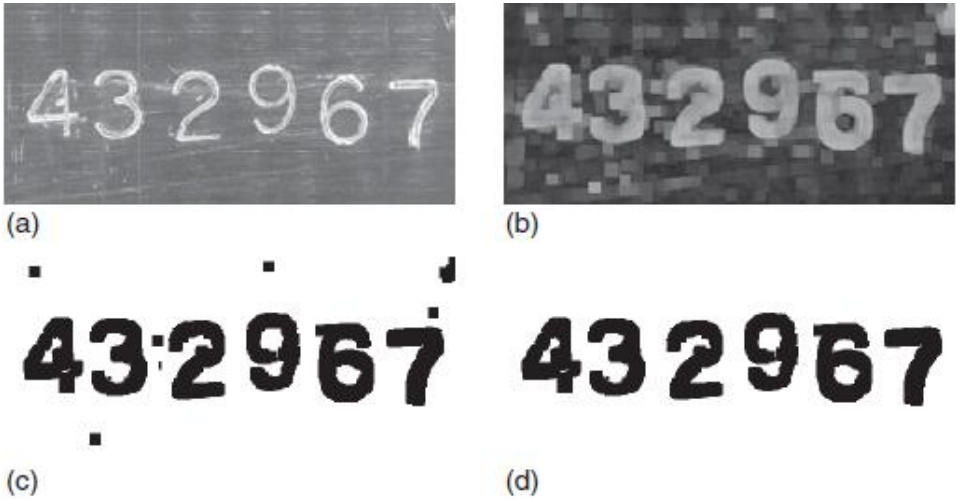
(c) Result of performing a gray value closing with an octagon of diameter 11 on (a).

(d) Result of segmenting the errors in (a) by using a dynamic threshold (*) operation with the images of (b) and (c).

(*) Dynamic thresholding:
is the operation of comparing the image to its local background.
The dynamic thresholding operation for bright objects is given by
$S = \{(r, c) \in R \mid f_{r,c} - g_{r,c} \geq thr\}$
while the dynamic thresholding operation for dark objects is given by
$S = \{(r, c) \in R \mid f_{r,c} - g_{r,c} \leq -thr\}$

Adapted from Handbook of Machine and Computer Vision
A. Hornberg (editor)

Morphological operators: Gray scale images



(a) Image showing a punched serial number. Because of the scratches, texture, and illumination, it is difficult to segment the characters directly.

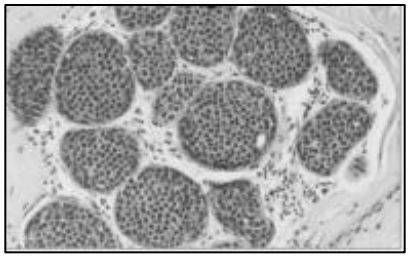
(b) Result of computing the morphologic gradient (difference between dilation and erosion) with a 9 × 9 square.

(c) Result of thresholding (b).

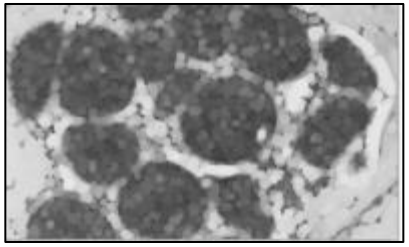
(d) Result of computing the connected components of (c) and selecting the characters based on their size.

Adapted from Handbook of Machine and Computer Vision
A. Hornberg (editor)

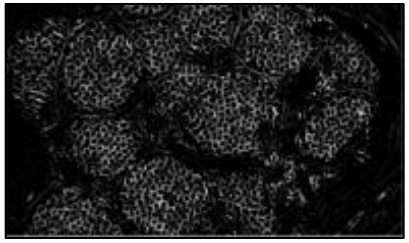
Morphological operators: Gray scale images



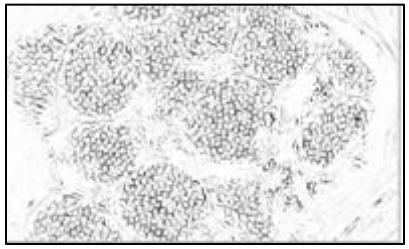
Original image



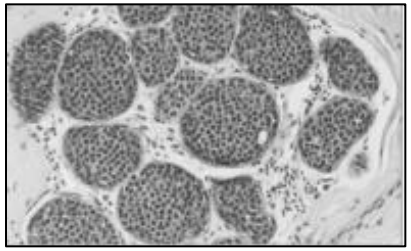
Opening



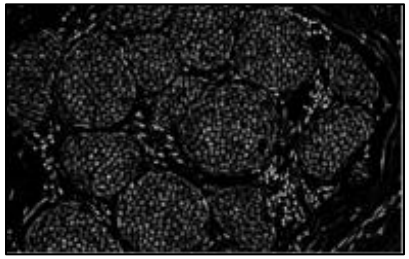
Top hat =
= Img – open(Img)



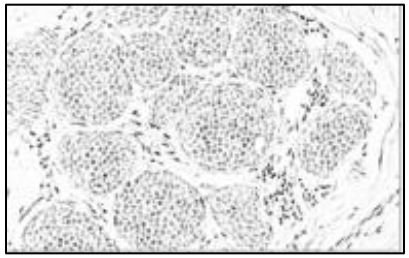
Top hat complemented



Closing



Black hat =
close(Img) - Img



Black hat complemented

http://www.cyto.purdue.edu/cdroms/micro2/content/education/wirth08.pdf